

DOSSIER PROJET

Développeur Web et Web Mobile

NGUYEN Thi Van Anh

Assofac - Formateur SOUMARE



Remerciements

Je tiens à exprimer ma sincère gratitude à toutes les personnes qui m'ont soutenu tout au long de cette formation.

Un grand merci à mon formateur, Monsieur **Demba Soumare**, pour sa patience, sa bienveillance et la confiance qu'il m'a accordée. Grâce à lui, j'ai pu bénéficier d'une expérience enrichissante lors de mon stage en développement backend chez Wyze Academy. Travailler avec des

technologies telles que React, Node.js et Express m'a permis de renforcer mes compétences pratiques dans un environnement professionnel.

Je tiens également à remercier **mes camarades** de classe pour leur esprit d'entraide et **ma famille** pour leur soutien moral, leur patience et leur compréhension, sans lesquels je n'aurais pas pu m'investir pleinement dans cette formation.

À vous tous, un immense merci pour votre contribution précieuse à cet accomplissement.

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

1

Sommaire

Remerciements.....	
1	
Sommaire.....	
2 Introduction.....	
3	
Résumé du projet.....	4
Compétences du référentiel couvertes.....	5
Cahier des charges.....	6
1. Contexte du Projet.....	6
2.	

Spécifications Fonctionnelles.....	6	3.
Spécificité technique (Environnement de développement).....	8	4.
Outils et technologies utilisés.....	9	5.
Charte graphique et logo.....	10	
Gestion de Projet.....	11	
1. Méthode Agile avec Trello.....	11	2.
Gestion du code source.....	12	3.
Maquettes.....	12	4.
Conception et modélisation des données.....	13	4.1. Un
dictionnaire de données.....	13	4.2.
MERISE.....	16	4.3.
Modélisation avec UML.....	18	
Développement de l'application.....	21	
1. Développement avec Docker et Docker Compose.....	21	2.
Intégration et démarrage du projet Symfony avec Docker.....	22	3.
Gestion des Données.....	23	4.
Accès aux données NoSQL avec Firebase.....	27	5.
Gestion des Routes, Contrôleurs et Formulaires.....	28	6.
Conception CRUD.....	31	7.
Conception Frontend.....	32	8.
Stratégie de Tests et Validation.....	34	9.
Sécurité du projet.....	36	10.
Gestion du Déploiement et Workflow CI/CD.....	39	11.
Processus de recherche et traduction.....	42	
Conclusion.....	4	
		3
Annexes.....	4	
		4

Introduction

Doté d'un esprit logique, d'une organisation rigoureuse et d'une grande curiosité, mon parcours académique m'a permis de développer des compétences analytiques solides. Ces qualités ont été déterminantes dans ma découverte du développement web, un domaine qui m'a rapidement captivé et qui correspond parfaitement à mes aptitudes et mes passions.

Bien que récent dans ce secteur, je suis profondément passionné par le développement web. Si j'apprécie la créativité du front-end, c'est la programmation back-end qui me fascine le plus, par sa complexité et son rôle central dans la conception de systèmes performants et sécurisés.

Convaincu d'avoir trouvé ma voie, je suis pleinement déterminé à exceller dans ce domaine et à contribuer à des projets innovants.

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

3

Résumé du projet

Dans le cadre du projet **Zoo Arcadia**, j'ai conçu et développé une application web complète pour **moderniser la visibilité** du zoo en ligne et améliorer l'expérience des visiteurs. Le zoo, situé près de la forêt de Brocéliande en Bretagne, souhaitait valoriser ses animaux, leur état de santé et ses services tout en optimisant la gestion interne pour les administrateurs, vétérinaires et employés.

J'ai créé des interfaces réactives avec **HTML**, **CSS** (avec **Bootstrap**), et j'ai développé le back-end avec **Symfony** et **PostgreSQL**. J'ai également conçu des outils spécifiques pour l'administration, facilitant la gestion des comptes, des services, des habitats, des animaux et des tâches quotidiennes des employés. **Firebase** a été intégré pour permettre une gestion des tâches en temps réel.

Le projet a été optimisé avec Webpack, déployé via **Docker** et **Docker Compose**, et automatisé grâce à un pipeline **CI/CD** avec **GitHub Actions**.

Le respect des normes de sécurité, d'accessibilité et de SEO a été une priorité, et des tests ont été réalisés avec **PHPUnit**, **Postman** et **MailDev**. Le monitoring a été mis en place pour assurer la performance et la fiabilité de l'application.

Ce projet m'a permis de renforcer mes compétences en développement **full-stack**, de la création des interfaces à la gestion des données, tout en respectant les valeurs écologiques du zoo et en apportant une solution à la fois fonctionnelle et respectueuse de l'environnement.

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

4

Compétences du référentiel couvertes

Dans le projet *Zoo Arcadia*, j'ai développé une application web full-stack en appliquant les compétences clés de ma formation :

Développer la partie front-end d'une application web ou web mobile sécurisée 1.

Installation et configuration de l'environnement de travail : J'ai configuré un environnement cohérent avec Docker, Docker Compose et Visual Studio Code, tout en automatisant l'intégration et le déploiement connus grâce à GitHub Actions.

2. **Maquage des interfaces utilisateur** : J'ai réalisé des maquettes interactives avec Figma pour valider rapidement les interfaces utilisateur et obtenir des retours itératifs.
3. **Création d'interfaces utilisateur statiques** : Les pages statiques ont été développées en HTML5 et CSS3. J'ai utilisé Bootstrap pour garantir une mise en page réactive et adaptée aux différents supports.
4. **Développement de la partie dynamique des interfaces utilisateur** : J'ai ajouté des fonctionnalités interactives.

Développer la partie back-end d'une application web ou web mobile sécurisée 1.

- Mise en place d'une base de données relationnelle** : J'ai modélisé la base de données avec l'approche Merise, produisant des diagrammes clairs et un dictionnaire de données détaillé.
2. **Développement des composants d'accès aux données SQL et NoSQL** : J'ai utilisé PostgreSQL pour les données relationnelles et Firebase pour les données non structurées, assurant une gestion des données flexible et performante.
 3. **Développement des composants métier côté serveur** : Structuration du back-end avec Symfony et Doctrine pour l'accès aux données et Nginx pour sécuriser l'application.
 4. **Documentation du déploiement** : J'ai produit une documentation détaillée pour le déploiement via Docker, Docker Compose et GitHub Actions, garantissant un déploiement sûr.

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

5

Cahier des charges

1. Contexte du Projet

Le Zoo Arcadia, situé près de la forêt de Brocéliande en Bretagne, fondé en 1960, est reconnu pour son engagement écologique et la qualité des soins apportés à ses animaux. Il abrite une grande variété d'animaux répartis dans différents habitats tels que la savane, la jungle et les marais. La santé et le bien-être des animaux sont des priorités, avec des contrôles vétérinaires quotidiens et une gestion alimentaire stricte. Le zoo bénéficie d'une excellente réputation et d'une solide stabilité financière.

Afin de moderniser ses opérations et d'améliorer sa visibilité en ligne, le zoo souhaite développer une application web pour valoriser ses animaux et les soins qu'ils reçoivent, tout en améliorant l'expérience des visiteurs.

Cette plateforme facilitera également la gestion interne du zoo, notamment pour les administrateurs, les employés et les vétérinaires.

Objectifs du Projet

- **Améliorer la visibilité en ligne** : Créer une application web éco responsable pour mettre en valeur les animaux du zoo et les soins qu'ils reçoivent.
- **Offrir une meilleure expérience utilisateur** : Permettre aux visiteurs de consulter facilement des informations sur les animaux, les services, les horaires et les habitats.
- **Optimiser la gestion interne du zoo** : Développer des outils facilitant la gestion des soins vétérinaires, des services visiteurs et des tâches administratives.

2. Spécifications Fonctionnelles

US 1 : Page d'accueil

Utilisateur concerné : Visiteur

- Présentation du zoo avec des images.
- Affichage des habitats, services et animaux du zoo.
- Selon l'avis des visiteurs.

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

6

US 2 : Menu de l'application

Utilisateur concerné : Visiteur

- Navigation facilitée : Accueil, services, habitats, connexion (réservée aux vétérinaires, employés, et administrateurs), contact.

US 3 : Vue globale de tous les services

Utilisateur concerné : Visiteur

- Récapitulatif des services (restauration, visites guidées, pet train)
- Description et configuration des services par l'administrateur

US 4 : Vue globale des habitats

Utilisateur concerné : Visiteur

- Liste des habitats avec images et noms.
- Détails sur chaque habitat, incluant la liste des animaux présents et leurs informations (prénom, race, état de santé, nourriture, soins, etc.).
- Accès aux rapports vétérinaires, incluant les informations sur l'état de l'animal, la nourriture, et les soins fournis.

US 5 : Avis

Utilisateur concerné : Visiteur

- Les visiteurs peuvent laisser un avis avec un pseudo et un texte, soumis à validation par un employé avant publication.

US 6 : Espace Administrateur

Utilisateur concerné : Administrateur

- Création de comptes Employé et Vétérinaire (email et mot de passe)
- Envoi de **lien de réinitialisation** de mot de passe pour première connexion -
- Gestion des services, habitats, animaux, et horaires
- Consultation des rapports vétérinaires
- Dashboard des consultations par animal
- **Aucune création de compte Administrateur via l'application**

US 7 : Espace Employé

Utilisateur concerné : Employé

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

7

- Validation/invalidation des avis.
- Modification des services du zoo.
- Gestion de l'alimentation des animaux: enregistrement de l'animal, date, nourriture et quantité.

US 8 : Espace Vétérinaire

Utilisateur concerné : Vétérinaire

- Remplir des comptes rendus pour chaque animal (état, nourriture donnée, date, soins). -
- Ajouter des commentaires sur l'état des habitats.
- Accéder à l'historique des alimentations des animaux fourni par les employés. **US**

9 : Connexion

Utilisateur concerné : Administrateur, vétérinaire, employé

- Connexion via email et mot de passe.

- Accès limité aux utilisateurs autorisés (Administrateur, Vétérinaire, Employé). **US**

10 : Contact

Utilisateur concerné : Visiteur

- Formulaire de contact pour poser des questions (titre, description, email). -
- Réponse par email d'un employé.

US 11 : Stasque sur la consultation des habitats

Utilisateur concerné : Visiteur

- Suivi des consultations des animaux via clics.
- Données stockées dans une **base non relationnelle**.
- Accès aux stasques par l'administrateur via un dashboard.

3. Spécificité technique (Environnement de développement)

- **Front-end** : HTML5, CSS (Bootstrap), JavaScript, npm et node.js, Twig
- **Back-end** : PHP 8.2, Composer, Symfony 7.1, EasyAdmin, Doctrine ORM, Nginx (HTTPS) -
- Base de données** : PostgreSQL avec pgAdmin, Firebase (NoSQL)
- **Outils de déploiement et monitoring** : Docker, Docker Compose, GitHub Actions (CI/CD), Grafana, Prometheus
- **Tests** : Postman, PHPUnit, MailDev

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

8

4. Outils et technologies utilisés

❖ Outils de Gestion du Projet et de Conception



En ce qui concerne la gestion du projet, **Trello** a été utilisé pour suivre l'avancement des tâches et assurer une gestion claire des priorités et des échéances.

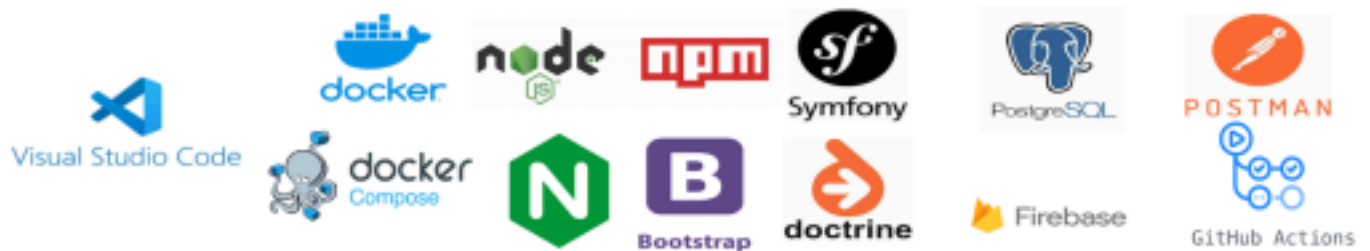
La gestion du code source s'est faite via **Git et GitHub**, permettant de centraliser les versions et de faciliter l'intégration continue.

Figma a servi à la création des maquettes et des prototypes de l'interface utilisateur, assurant ainsi une validation rapide des designs.

Looping et **Lucidchart** ont servi à la modélisation des processus et des flux de données selon la méthodologie MERISE, ce qui a permis une structuration efficace des données.

Pour la gestion documentaire, **Google Drive** a centralisé la gestion documentaire, tandis que **Canva** a été utilisé pour le logo et **Pexels** pour les images libres de droits.

❖ Configuration et Développement Technique



Pour garantir un environnement stable et sécurisé, j'ai utilisé **Visual Studio Code** comme IDE principal pour sa flexibilité et ses extensions compatibles avec les langages utilisés, tels que PHP, JavaScript, et HTML/CSS.

Docker et Docker Compose ont permis de conteneuriser et d'isoler les services, offrant un environnement portable pour le backend et la compilation frontend.

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

9

Les dépendances frontend, comme **Bootstrap** et **SCSS**, ont été gérées avec **npm (Node.js)**, tandis que **Webpack Encore** a permis d'optimiser et de regrouper les fichiers CSS et JavaScript.

Nginx a été configuré comme proxy et pour sécuriser les connexions, tandis que **Symfony** (backend) et **Doctrine** ORM ont facilité la gestion des données relationnelles dans **PostgreSQL**. J'ai intégré **Firebase** pour gérer les données non structurées et les images en temps réel.

GitHub Actions m'ont permis d'automatiser les tests, les intégrations garantissant une gestion fluide des mises à jour.

Enfin, avec **Postman**, j'ai testé et validé les API, garantissant une communication fluide entre le frontend et le backend.

Ces outils ont créé un environnement efficace et performant pour le projet.

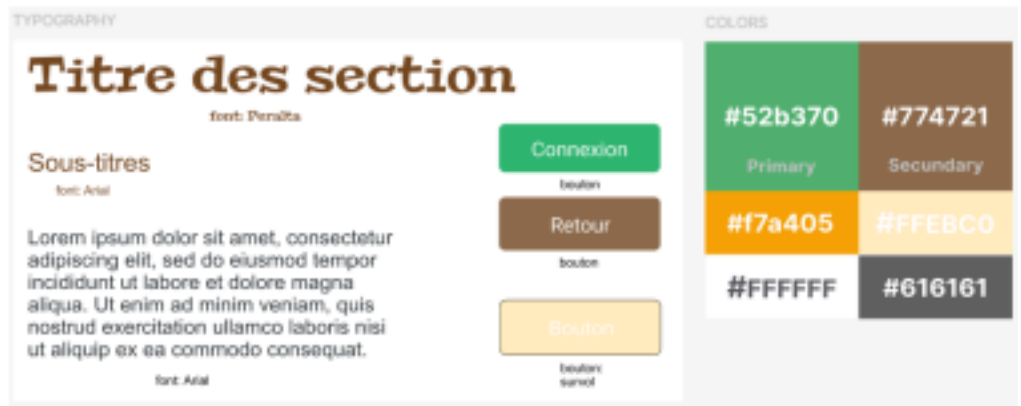
5. Charte graphique et logo

Le logo est conçu avec des branches et feuilles entourant le mot "Zoo", symbolisant la nature et

la protection des animaux.

J'ai opté pour une **palette de couleurs inspirée de la nature** pour créer une atmosphère fraîche et naturelle.

Les polices modernes et facilement lisibles apportent une touche de simplicité et de convivialité à l'interface de l'application.



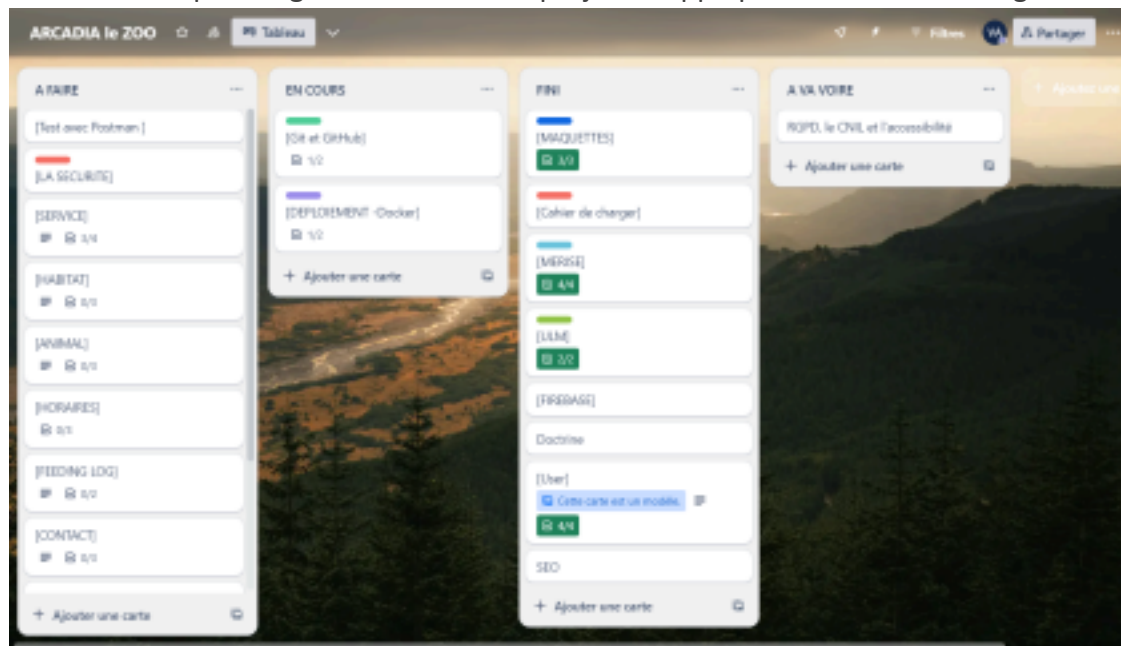
Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

10

Gestion de Projet

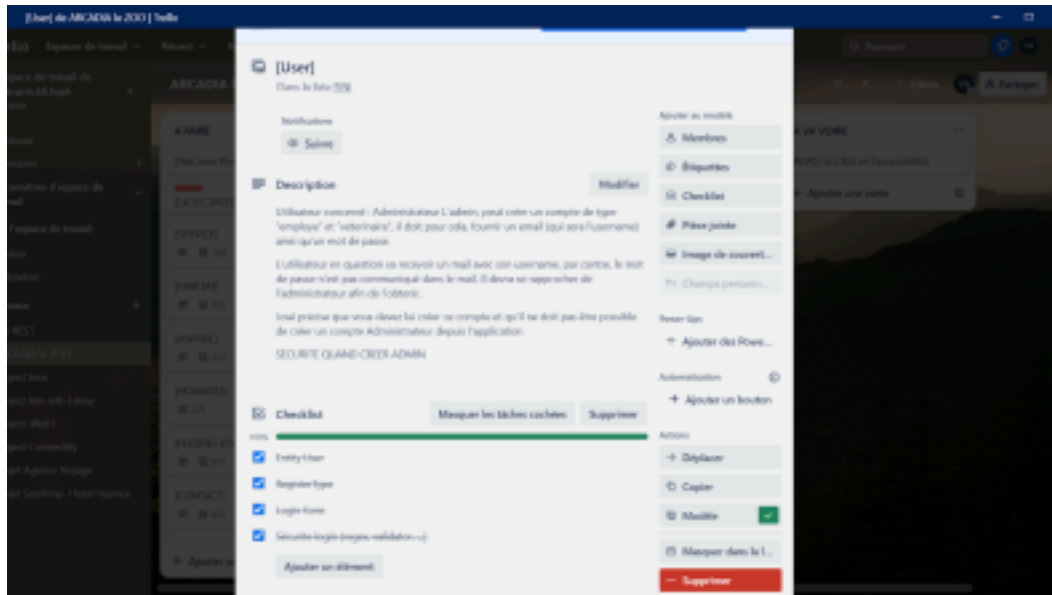
1. Méthode Agile avec Trello

J'ai utilisé Trello pour organiser et suivre le projet en appliquant une méthode Agile.



Les tâches étaient organisées en colonnes (« À FAIRE, EN COURS, FINI, À VOIR ») et étagées par type, avec des descripteurs et une checklist. Cela a permis un suivi clair et structuré tout au long

du projet.



Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

11

2. Gestion du code source

J'ai utilisé Git et GitHub pour gérer le code source, en créant plusieurs branches pour chaque fonctionnalité. Une fois les fonctionnalités testées et validées, je les fusionne avec la branche principale (main) pour garantir sa stabilité. J'ai également utilisé des **tags** pour marquer les versions stables. Un fichier README.md a été ajouté pour documenter l'utilisation du projet.

Voici les commandes Git que j'utilise le plus souvent :

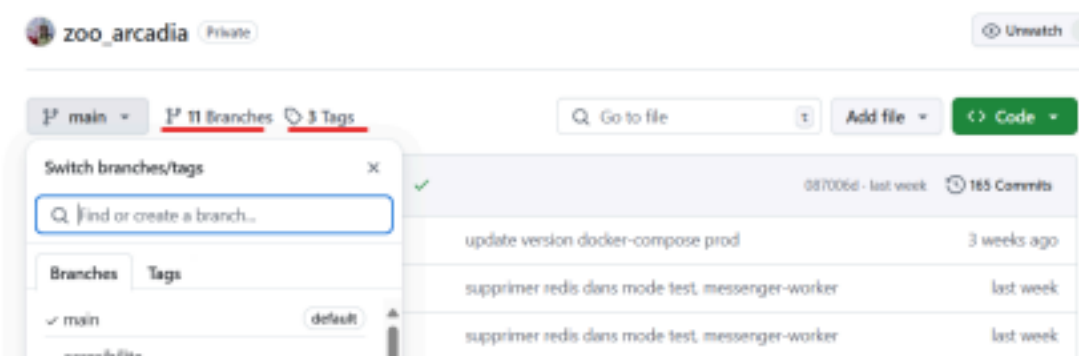
git checkout -b [nom_branche] : Créer et se déplacer vers une nouvelle branche.

git merge [nom_branche] : Fusionner une branche dans la branche principale.

git status : Vérifier l'état des modifications.

git add : Ajouter des fichiers modifiés à l'index.

git commit -m "[message]" : Valider les modifications avec un message descriptif.



push : Pousser les modifications vers GitHub.

git tag [nom_tag] : Marquer une version stable du projet.

3. Maquettes

J'ai conçu des maquettes interactives sur Figma, incluant zoning, wireframes, et prototypes pour desktop et mobile. Les interfaces ont été adaptées aux visiteurs et administrateurs, assurant une navigation fluide et une gestion efficace des informations. **(Voir Annexe 1 - Page 44)**

4. Conception et modélisation des données

4.1. Un dictionnaire de données

Mnémonique	Type	Contrainte	Description
User			
id_user	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Identifiant unique de l'utilisateur.
email	VARCHAR(255)	UNIQUE, NOT NULL	Adresse email de l'utilisateur.
password	VARCHAR(255)	NOT NULL	Mot de passe crypté.
roles	JSON	NOT NULL	Rôles de l'utilisateur (admin, vétérinaire, employé).
createdAt	DATETIME	NOT NULL	Date et heure de création.
Service			
id_service	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Identifiant unique du service.

		T	
name	VARCHAR(255)	NOT NULL	Nom du service.
images	JSON	NOT NULL	Liste des URLs des images du service.
descripon	TEXT	NOT NULL	Descripon détaillée du service.
Habitat			
id_habitat	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Identifiant unique de l'habitat.
name	VARCHAR(255)	NOT NULL	Nom de l'habitat.
images	JSON	NOT NULL	Liste des URLs des images de l'habitat.
descripon	TEXT	NOT NULL	Descripon détaillée de l'habitat.
Animal			
id_animal	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Identifiant unique de l'animal.
name	VARCHAR(255)	NOT NULL	Nom de l'animal.

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

13

species	VARCHAR(255)	NOT NULL	Espèce ou race de l'animal.
images	JSON	NOT NULL	Liste des URLs des images de l'animal.
FeedingLog			
id_feeding	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Identifiant unique de la consommaon de nourriture
date_feeding	DATETIME	NOT NULL	Date et heure de l'alimentaon
food_given	VARCHAR(255)	NOT NULL	Type de nourriture donnée à l'animal
food_quanty	DECIMAL(10,2)	NOT NULL	Quanté de nourriture donnée (en grammes)
VeterinarianReport			
id_report	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Identifiant unique du compte-rendu vétérinaire.

condion_animal	VARCHAR(255)	NOT NULL	État de l'animal
food_given	VARCHAR(255)	NOT NULL	Type de nourriture proposée à l'animal.
food_quanty	DECIMAL(10,2)	NOT NULL	Quanté de nourriture en grammes.
condion_detail	TEXT	NULL	Détail supplémentaire sur l'état de l'animal.
date_visit	DATETIME	NOT NULL	Date et heure de la visite.
CommentHabitat			
id_comment	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Idenfiant unique du commentaire.
comment	TEXT	NOT NULL	Contenu du commentaire.
condionHabitat	VARCHAR(255)	NOT NULL	État actuel de l'habitat.
improvement	BOOLEAN	NOT NULL	Indicateur d'amélioraon nécessaire.
createAt	DATETIME	NOT NULL	Date et heure de créaon du commentaire.
Contact			
id_contact	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Idenfiant unique de la demande de contact.
email	VARCHAR(255)	NOT NULL	Adresse email de l'expéditeur.

subject	VARCHAR(255)	NOT NULL	Titre de la demande.
message	TEXT	NOT NULL	Descripon détaillée de la demande.
status	VARCHAR(150)	NOT NULL	Statut de la demande (ex: Pending, Replied, ...).
createAt	DATETIME	NOT NULL	Date et heure de créaon.
Review			
id_review	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Idenfiant unique de l'avis.
pseudo	VARCHAR(255)	NOT NULL	Pseudonyme de l'utilisateur.
review	TEXT	NOT NULL	Contenu de l'avis.

approved	BOOLEAN	NOT NULL	Statut de validaton de l'avis (true/false).
createAt	DATETIME	NOTNULL	Date et heure de créaon de l'avis.
Schedule			
id_schedule	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Idenfiant unique du planning
dayOfWeek	VARCHAR(255)	NOT NULL	Jour de la semaine (ex : lundi, mardi).
openingTime	TIME	NOT NULL	Heure d'ouverture du zoo.
closingTime	TIME	NOT NULL	Heure de fermeture du zoo.
isHoliday	BOOLEAN	NULL	Indicateur si c'est un jour férié.
ResetPasswordRequest			
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Idenfiant unique de la demande.
selector	VARCHAR(20)	NOT NULL	Sélecteur pour la demande de réinitialisaon.
hashedToken	VARCHAR(100)	NOT NULL	Token haché pour la réinitialisaon.
requestedAt	DATETIME	NOT NULL	Date et heure de la demande de réinitialisaon.
expiresAt	DATETIME	NOT NULL	Date et heure d'expiraon du token.

4.2. MERISE

4.2.1. MCD (Modèle Conceptuel de Données)

Le MCD idenfie les **entés** principales, leurs **atributs** et les **relaons** entre elles, en précisant leurs **cardinalités**. Par exemple, la relaon **Habitat (1, N) - Animal (1, 1)** indique qu'un habitat peut contenir plusieurs animaux, mais chaque animal appartient à un seul habitat. De même, la relaon **Users (1, N) - Service (1, N)** montre qu'un utilisateur peut gérer plusieurs services, et un service peut être géré par plusieurs utilisateurs. Ces relaons conceptuelles reflètent les besoins méers tout en assurant une modélisaon logique.

4.2.2. MLD (Modèle Logique de Données)

Le MLD traduit les entités, attributs, et relations du MCD en une structure relationnelle composée de **tables**, **champs** (colonnes), **clés primaires**, et **clés étrangères** pour représenter les

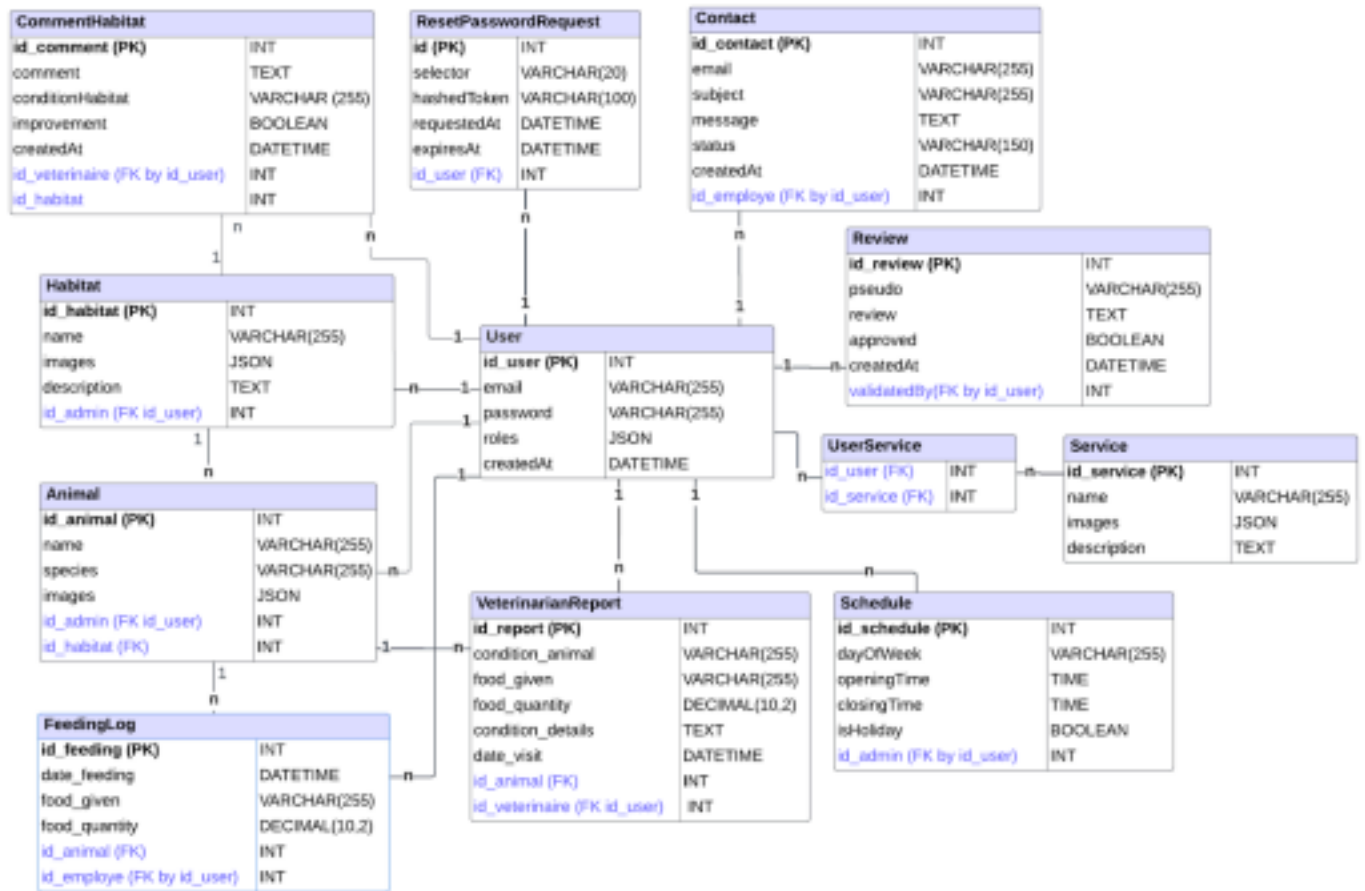
Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

16

associations. Par exemple, une association **many-to-one** entre Animal et Habitat est réalisée par l'ajout d'une clé étrangère **id_habitat** dans la table Animal, permettant à plusieurs animaux d'appartenir à un même habitat. Les associations **many-to-many**, comme entre Users et Service, sont gérées par des **tables intermédiaires** (UserService) contenant deux **clés étrangères** (id_user, id_service). Chaque table inclut ses champs définis, tels que **id_user**, **email**, et **roles** pour la table **Users**, pour garantir une structure prête à l'implémentation.

4.2.3. MPD (Modèle Physique de Données)

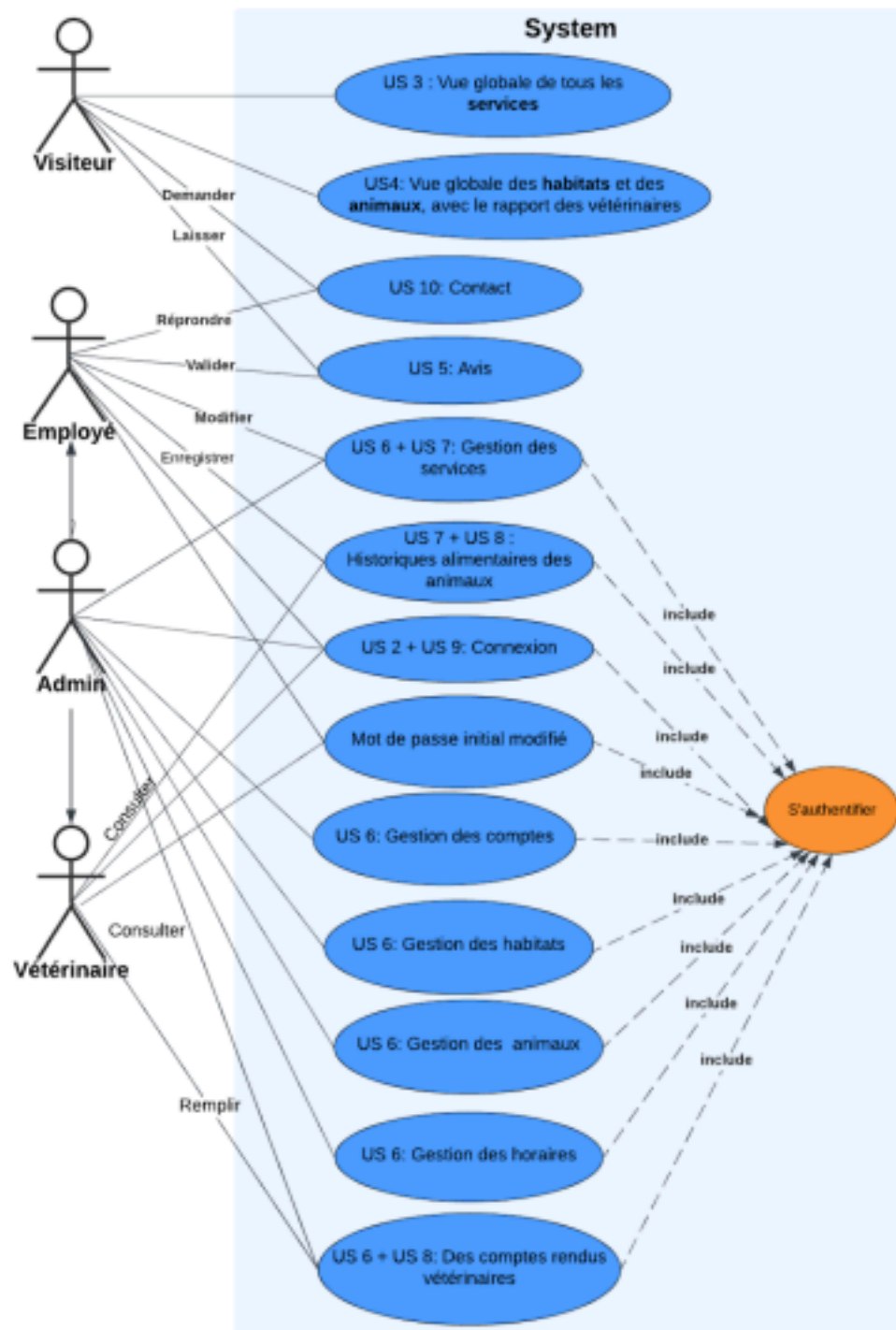
Le MPD précise les **tables**, **colonnes**, **types de données** et **contraintes**. Par exemple, la table **Users** contient email (VARCHAR(255)) et roles (JSON). Les clés étrangères assurent l'intégrité référennelle. Les associations **many-to-many** restent gérées par des tables comme UserService pour optimiser les performances en base.



4.3. Modélisation avec UML

4.3.1. Diagrammes de cas d'utilisation

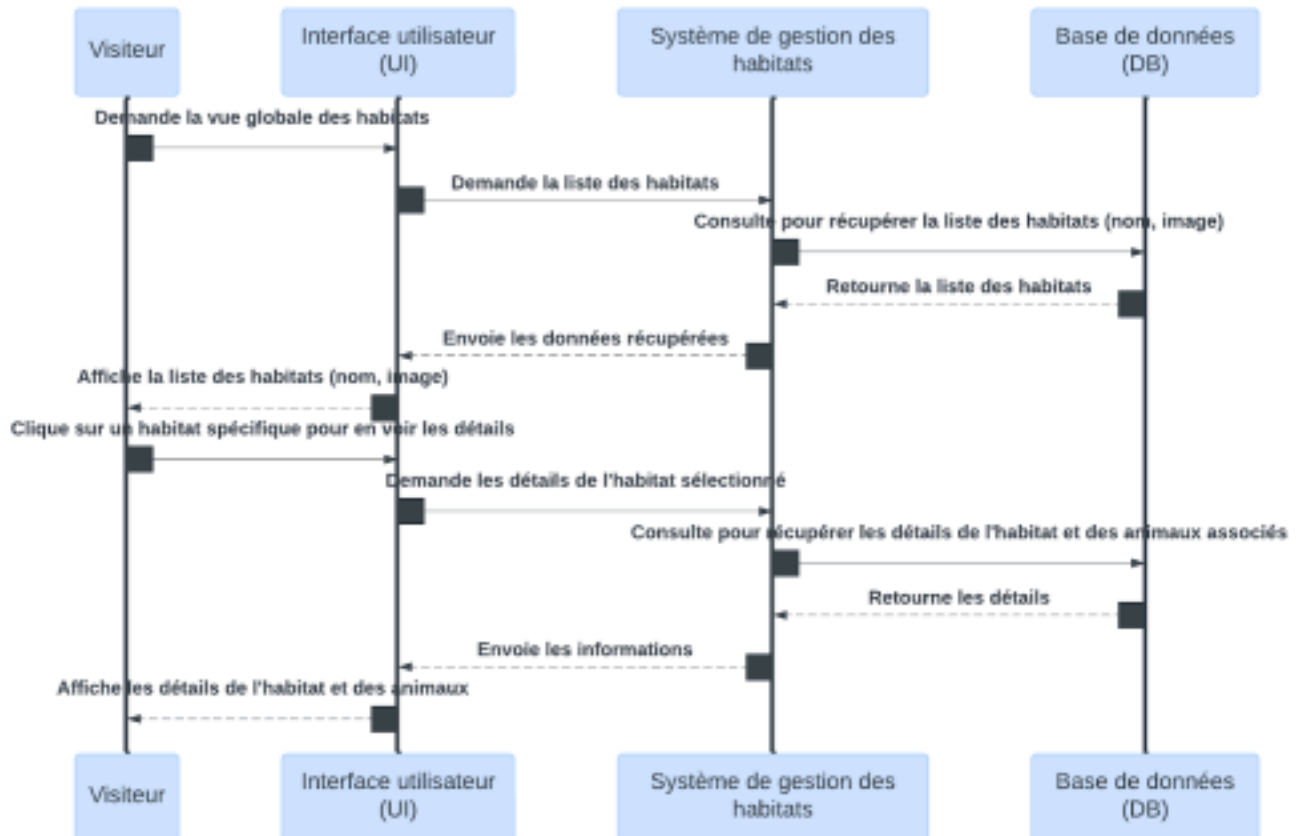
Ce diagramme illustre les interactions entre les acteurs clés (visiteurs, administrateurs, employés, vétérinaires) et le système, avant de définir les fonctionnalités. Il clarifie les responsabilités de chaque utilisateur et offre une vue globale des processus.



4.3.2. Diagramme de Séquence

Les diagrammes de séquence montrent l'enchaînement des interactions entre les acteurs et le système dans des scénarios spécifiques.

Le visiteur consulte la liste des habitats via l'UI, qui interroge le système de gestion pour récupérer et afficher les données depuis la base de données. En sélectionnant un habitat, l'UI demande des détails supplémentaires (habitat et animaux), que le système fournit après interrogation de la base de données.



❖ Connexion

L'utilisateur saisit ses identifiants via l'UI, qui transmet la requête au système d'authentification. Après vérification dans la base de données, un message d'erreur est affiché en cas d'échec, ou une redirection vers l'espace correspondant (Administrateur, Employé, Vétérinaire) est effectuée en cas de succès. (**Voir Annexe 2 - Page 49**)

1. Développement avec Docker et Docker Compose

Pour garantir un environnement homogène, reproductible et sécurisé, j'ai choisi Docker et Docker Compose.

❖ Dockerfile

Pour le service PHP, j'ai créé un Dockerfile personnalisant l'image avec PHP 8.2, les extensions nécessaires (pdo_pgsql), Composer pour la gestion des dépendances, Symfony CLI pour les commandes Symfony, et Node.js pour la gestion des assets avec Webpack Encore.

Voir Annexe 3 - Page 49

❖ Docker Compose

Le fichier docker-compose.yml orchestre des services essentiels (PostgreSQL, Nginx, PHP, Prometheus,...) en utilisant des **images** Docker pour créer des **conteneurs**. Les **volumes** assurent la persistance des données, tandis qu'un **réseau** personnalisé isole les services et permet leur communication sécurisée.

Les variables d'environnement sont définies dans le fichier .env, permettant de personnaliser les configurations des services. (Voir Annexe 3)

Processus de construction des conteneurs avec Docker Compose

```
PS C:\Users\user\Desktop\Documents\zoo-artadia\docker> docker-compose up --build -d

[*] Building 292-66 (36/16) F1039ED
=> [php_service Internal] load build definition from dockerfile
=> => transferring dockerfile: 1.05kB
=> [php_service Internal] load metadata for docker.io/library/php:8.2-fpm
=> [php_service Internal] load metadata for docker.io/library/composer:latest
=> [php_service Internal] load .dockerignore
=> => transferring context: 2kB
=> [php_service] FROM docker.io/library/composer:latest@sha256:e6c1ac329256c25b8dee572df334900570fb26b6baaa788abe69b64181701e1
=> => resolve docker.io/library/composer:latest@sha256:e6c1ac329256c25b8dee572df334900570fb26b6baaa788abe69b64181701e1
=> => sha256:e6c1ac329256c25b8dee572df334900570fb26b6baaa788abe69b64181701e1 18.0kB / 18.0kB
=> => sha256:a3d88d1cdd17c5d8e77c75a19f12c7a8574bda1677aeaf3109e7e6126a2988 11.6kB / 11.6kB
=> => sha256:0af409e61285d880321645498daf2c381b36852f1a8b898d8b730831a2d3ea 3.61kB / 3.61kB
=> => sha256:1f3e9299c25666fa2846e76c3748b732c2d8d61479c244032b32134f9 3.64kB / 3.64kB
=> => extracting sha256:1f3e9299c25666fa2846e76c3748b732c2d8d61479c244032b32134f9
=> => extracting sha256:1f3e9299c25666fa2846e76c3748b732c2d8d61479c244032b32134f9
```

Les services sont gérés par Docker Compose avec les commandes suivantes :

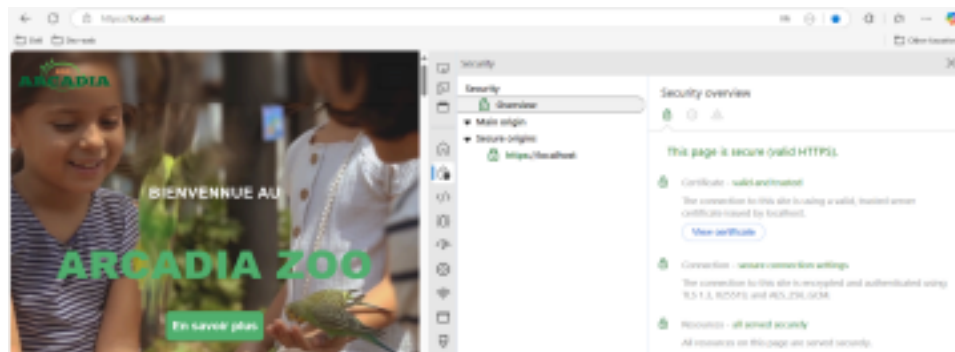
- Démarrage des conteneurs : **docker-compose up --build -d**
- Accès au conteneur PHP : **docker exec -it php_server bash**
- Démarrage en arrière-plan : **docker-compose up -d**
- Arrêt des conteneurs : **docker-compose down**

Grâce à ces outils et commandes, la gestion de l'application est simplifiée, tout en garantissant un environnement cohérent et fiable.

❖ Sécurisation des connexions avec Nginx

J'ai mis en place un **certificat SSL auto-signé** avec Nginx afin de garantir des connexions sécurisées en local. Ce certificat ainsi que la configuration Nginx ont été intégrés dans le fichier `docker-compose.yml` pour rediriger les requêtes HTTP vers HTTPS et ajouter des en-têtes de sécurité, protégeant ainsi l'application contre les attaques courantes.

Voir Annexe 4 : Configuration HTTPS - Page 50



2. Intégration et démarrage du projet Symfony avec Docker 2.1.

Architecture MVC dans Symfony

Pour structurer mon application, j'ai choisi Symfony pour son respect rigoureux de l'architecture MVC. Les contrôleurs gèrent les requêtes, la logique métier et orchestrent les interactions entre le modèle et la vue. Le modèle contient les entités (User, Habitat, Animal...) et la logique de données avec Doctrine ORM pour les opérations CRUD. La vue affiche les données via Twig, avec Bootstrap pour un design réactif.

2.2. Intégration et démarrage du projet Symfony avec Docker

- Initialisation et installation des dépendances

J'ai initialisé le projet Symfony en utilisant la commande `symfony new --webapp app`. Ensuite, j'ai accédé au conteneur PHP via `docker exec -it php_server bash` pour installer les dépendances essentielles avec Composer comme *symfony/security-bundle*, *doctrine/doctrine-migrations-bundle*, *easycorp/easyadmin-bundle*. Des dépendances pourront être ajoutées selon les besoins du projet.

- Configuration des variables d'environnement

J'ai configuré le fichier `.env` pour lier l'application aux services Docker (base de données PostgreSQL, Maildev, Firebase) garantissant une adaptabilité aux environnements de développement et de production sans modifications supplémentaires.

Installation du projet Symfony



Gestion des packages avec Composer



Le fichier .env

3. Gestion des Données

- ❖ Gestion des Données avec Doctrine et PostgreSQL

Pour structurer la base de données, j'ai utilisé Doctrine ORM, qui assure une gestion fluide des

données relationnelles.

Avec les configurations `.env` établies, la commande `php bin/console doctrine:database:create` a permis de créer la base de données.


Les entités ont été générées avec `php bin/console make:entity`, en définissant les caractéristiques de chaque champ (type, taille, nullable) ainsi que leurs relations (**ManyToOne**, **OneToMany**).

Pour synchroniser les entités avec la base, j'ai appliqué les migrations avec `php bin/console make:migration` et exécuté `php bin/console doctrine:migrations:migrate`.

Ensuite, pour maintenir la cohérence de la structure, j'ai régulièrement utilisé `php bin/console doctrine:schema:update`.

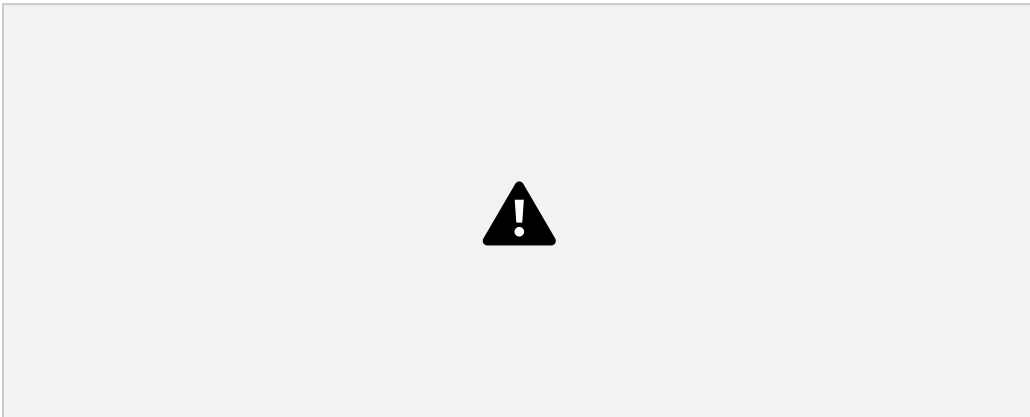
Enfin, pour vérifier l'état des migrations, j'ai exécuté la commande `php bin/console doctrine:migrations:status`.

Commande de création de la base de données

Création l'entité Animal et ses relations	Entité Animal
	



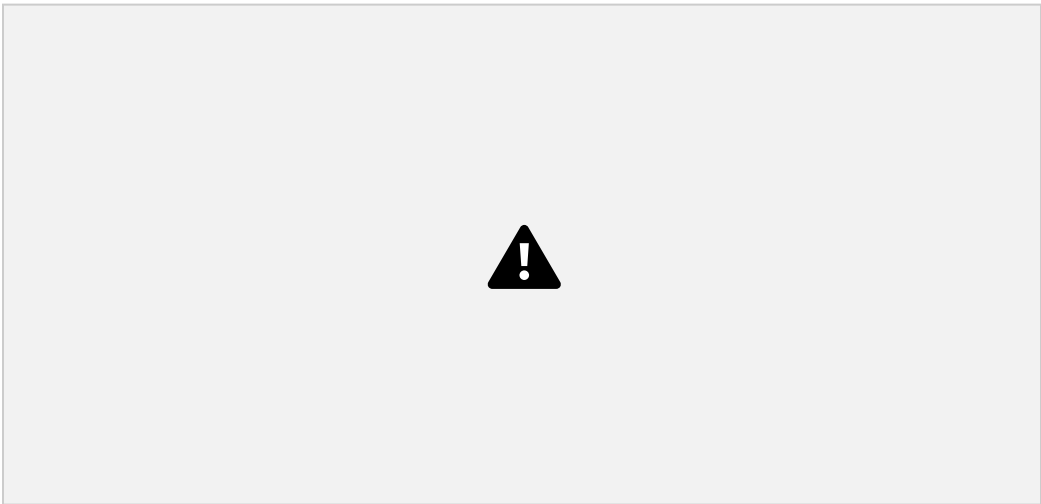
Migrations et Synchronisation



Mise à jour la base de donnée



Migration Status



Des requêtes SQL ont été utilisées pour insérer, mettre à jour et sélectionner des données dans la base PostgreSQL. Par exemple :

- **Inseron d'un nouvel animal (Jaguar):** La requête SQL ajoute un nouvel animal dans la table animal, avec ses détails tels que son nom, son habitat, etc.



- **Mise à jour d'un animal existant:**





Une requête de
son nom et sa race.

- **Sélecon des détails des animaux** : Des requêtes de sélecon récupèrent des informaons détaillées sur les animaux, y compris leur habitat et leur administrateur



4. Accès aux données NoSQL avec Firebase

J'ai intégré Firebase pour gérer les données en temps réel et le téléversement d'images. Le fichier **serviceAccount.json** est placé dans config/firebase, et la connexion a été configurée via **.env** et **services.yaml** après l'installaon de krait/firebase-php.



Un fichier **FirestoreService.php** centralise les fonctions appelées dans les contrôleurs, comme **AnimalCrudController** et **HabitatCrudController** pour le téléversement d'images (URLs enregistrées dans PostgreSQL). **AnimalController** met à jour en temps réel le compteur de vues des animaux. ([Voir Annexe 8](#) - Page 57)

Visualisaon dans Realm Database Graphique des vues par animal

Les URLs des images téléversées sont stockées dans la base de données PostgreSQL



5. Gestion des Routes, Contrôleurs et Formulaires

Après avoir défini les entités principales, j'ai configuré les routes et implémenté des contrôleurs pour chaque fonctionnalité. Les routes sont liées aux actions des contrôleurs, assurant une navigation fluide et une organisation logique, ce qui facilite la gestion, l'évolution et la maintenance de l'application.

• Système d'Authentification



Pour intégrer le formulaire d'authentification, j'ai installé *symfony/security-bundle* et généré la structure de base avec *symfony console make:security*. Cela a automatiquement créé les routes de connexion/déconnexion et un contrôleur (SecurityController).

Ensuite, j'ai personnalisé l'authentification avec une classe *LoginAuthenticator* basée sur *AbstractLoginFormAuthenticator*. Cela m'a permis de configurer des redirections dynamiques selon les rôles et d'ajouter des protections. ([Voir Annexe 6 - Page 52](#))

• Gestion de l'inscription et réinitialisation sécurisée des mots de passe



La gestion des comptes utilisateurs (employés, vétérinaires) est gérée via EasyAdmin.

Lors de l'inscription, l'administrateur attribue l'email et le rôle, générant un mot de passe temporaire haché.

Un lien sécurisé de **réinitialisation** est envoyé à l'utilisateur pour définir son mot de passe. Ce processus est pris en charge par le ResetPasswordBundle avec les commandes *"composer require symfonycasts/reset-password-bundle"* et *"php bin/console make:reset-password"*.

Il applique des règles strictes de complexité de mot de passe, génère des tokens temporaires à expiration rapide et renouvelle les sessions après chaque réinitialisation. Cela permet une gestion centralisée et sécurisée des mots de passe.

(Voir Annexe 7 - Page 55)

Créer nouvel utilisateur



lien de réinitialisation sécurisé

Email contenant le



● Formulaire de contact



J'ai créé un formulaire de contact avec php bin/console make:controller et php bin/console make:form, enregistrant les demandes avec le statut "Pending". L'envoi d'emails a été intégré via MailerInterface et testé en développement avec **Maildev** avant la producon.

(Voir Annexe 13 : Tests Fonconnels - Page 61)

● Soumere et valider les avis

J'ai créé un formulaire pour recueillir les avis des ulisateurs, enregistrant leur pseudo et contenu. Par défaut, les avis sont "non approuvés" et validés via l'interface d'administraon. Une fois validés, ils sont affichés sur le site via un repository personnalisé, assurant un contrôle qualité opmal.



Tableau de bord des avis



Page d'affichage des avis validés



6. Concepon CRUD

L'applicaon intègre des interfaces CRUD adaptées aux rôles des utilisateurs :

- ❖ **Administrateurs, Employés, Vétérinaires** : EasyAdmin a été intégré via composer require easycorp/easyadmin-bundle, facilitant la créaon de tableaux de bord et de CRUD avec symfony console make:admin:dashboard et make:admin:crud.
 - ❖ **Visiteurs** : Des routes et contrôleurs spécifiques leur permeent de consulter (Read) les informaons publiques et d'ajouter (Create) des avis ou des demandes de contact, garansant une interacon simple et sécurisée.
- Interface Admin



- Interface Employé - Interface Visiteur

(Voir Annexe 9 : CRUD Admin avec EasyAdmin - Page 58)

7. Concepon Frontend

❖ Technologies Utilisées : Webpack Encore et Bootstrap

J'ai intégré **Webpack Encore** (via `composer require symfony/webpack-encore-bundle`) pour optimiser les fichiers CSS et JS, améliorer les performances et assurer l'intégrité des fichiers grâce au hashing, avec **Node.js** gérant les dépendances via `package.json`. Les styles **SCSS** sont compilés avec `npm run dev`, et **Bootstrap** permet de créer des interfaces réacves.

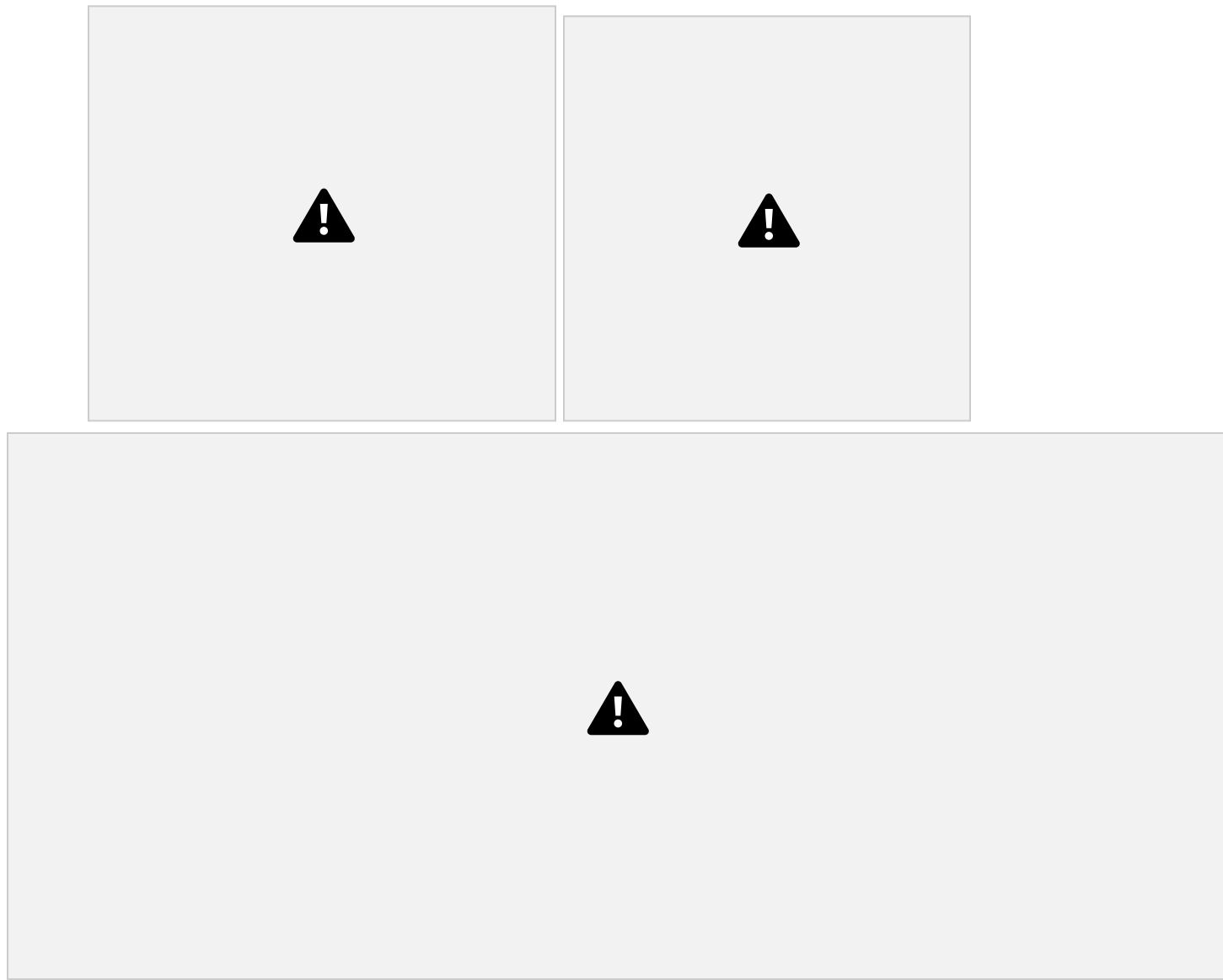
(Voir Annexe 10 : Configuraon Webpack - Page 59)

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

32

❖ HTML, CSS et Responsive Design

Le projet ulise une structure HTML sémantique pour améliorer l'accessibilité et le SEO. Les styles définissent une identité visuelle cohérente, et grâce à Bootstrap et aux media queries, le design est responsive et s'adapte à toutes les tailles d'écran.



❖ Deux Interfaces Utilisateurs : Visiteur et Dashboard (Voir Concepon CRUD - Arborescence - Page 31, 32)

❖ Optimisation SEO

Pour améliorer le référencement naturel (SEO), j'ai optimisé **la structure du site** avec des

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

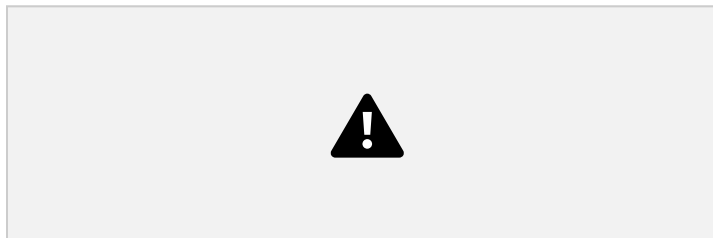
33

balises HTML sémantiques, intégré des **meta balises** (title, description) adaptées, et utilisé des **mots-clés** ciblés comme "Zoo Bretagne". **Les images** ont été optimisées avec des noms descriptifs et des **attributs alt**. Enfin, j'ai **compressé les fichiers CSS et JS** via Webpack pour accélérer le chargement et améliorer l'expérience utilisateur. (Voir Annexe 11 - Page 59)

❖ Accessibilité

Pour garantir une expérience inclusive, j'ai intégré plusieurs mesures d'accessibilité. Cela inclut des **labels** explicites pour les formulaires et boutons, des **attributs ARIA** pour les lecteurs d'écran,

un **contraste optimisé** pour la lisibilité, et des **textes alternatifs (alt)** pour les images. La structure utilise des **balises HTML sémantiques** pour une organisation logique. De plus, le site est **navisible au clavier** grâce aux **valeurs tabindex** attribuées aux éléments interactifs. ([Voir Annexe 5 : MVC - Page 51](#))



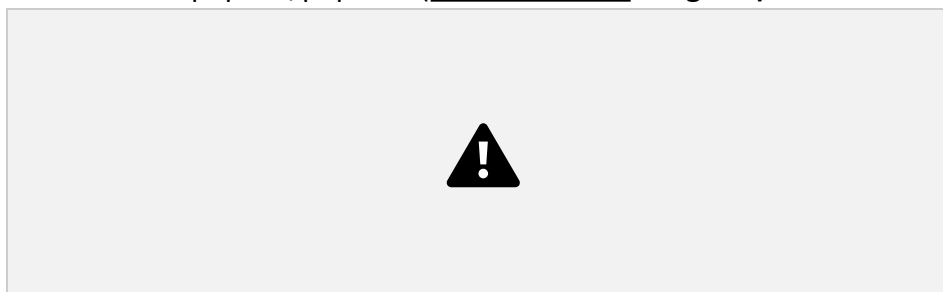
8. Stratégie de Tests et Validation

❖ Tests Unitaires avec PHPUnit

Pour garantir la fiabilité, j'ai configuré un environnement de test avec PHPUnit (composer require --dev phpunit/phpunit). Un fichier .env.test a été créé pour configurer une base de données PostgreSQL dédiée, avec migrations appliquées via APP_ENV=test symfony console doctrine:database:create et APP_ENV=test symfony console doctrine:migrations:migrate -n.

J'ai utilisé php bin/console make:test pour générer des tests unitaires (**TestCase**) et fonctionnels (**WebTestCase**), validant la sécurité et la logique métier des entités et contrôleurs.

Les tests sont exécutés avec php bin/phpunit. ([Voir Annexe 12 - Page 60](#))



Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

34

❖ Tests avec Postman

Des tests manuels ont été réalisés avec Postman pour vérifier les API REST et la sécurité : •

Validation du Token CSRF : Un token CSRF récupéré en **GET** est utilisé pour sécuriser les requêtes **POST** lors de l'authentification, protégeant contre les attaques CSRF.

- **Test de Limitation de Requêtes (Rate Limiting)** : Les requêtes répétées confirment que le serveur bloque les tentatives excessives de connexion..

GET Token CSRF



POST avec Token CSRF



Blocage Rate Liming

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

35

❖ Tests Fonconnels : Scénarios Réels

J'ai testé la configuraon de Maildev avec la commande `php bin/console mailer:test` et l'envoi de formulaire de contact depuis l'interface visiteur. Les données ont été vérifiées dans l'admin et la récepon des emails dans Maildev, garansant le bon fonconnement.

(Voir Annexe 13 - Page 61).

9. Sécurité du projet

La sécurité du projet repose sur plusieurs couches de protection, assurant une gestion rigoureuse des données sensibles et des interactions utilisateur.

• Routes et privilèges



Les accès sont sécurisés par des rôles (Admin, Employé, Visiteur). Par exemple, seuls les administrateurs gèrent les animaux et habitats, tandis que les visiteurs accèdent aux informations publiques.

(Voir Annexe 6 - Page 52)

Twing dashboard



Mise en place des rôles dans le fichier security.yaml et DashboardController



- **RGPD et CNIL** : L'application respecte pleinement le RGPD et la CNIL en garantissant la gestion transparente et sécurisée des données personnelles. Les **mots de passe** sont hachés et les **données sensibles** cryptées. Les administrateurs ont un accès en lecture seule aux avis et contacts pour préserver l'intégrité et la confidentialité des informations.

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

36

Les utilisateurs peuvent exercer leurs **droits d'accès**, de **rectification** et de **suppression**. Les données sont conservées pour une durée limitée et supprimées après cette période. Une politique de confidentialité claire est également disponible pour assurer la transparence. (**Voir Annexe 14 : RGPD & CNIL - Page 62**)

- **Sécurisation Serveur et Requêtes** : Les échanges entre le client et le serveur sont sécurisés via **HTTPS**, garantissant le chiffrement des données.

De plus, des en-têtes HTTP spécifiques sont configurés pour prévenir les attaques fréquentes telles que le **XSS**, le clickjacking et le MIME sniffing. La politique **HSTS** force l'utilisation du HTTPS, tandis que la **Content Security Policy (CSP)** limite les sources de contenu pour prévenir l'injection malveillante. Ces mesures assurent la confidentialité et l'intégrité des communications, renforçant ainsi la sécurité et la confiance des utilisateurs.

(Voir Annexe 4 : Configuration HTTPS - Page 50)

- **Base de données : Doctrine ORM** empêche les **injections SQL**. Les entrées utilisateur sont validées par des contraintes de validation Symfony afin de garantir que les données respectent les règles de format et de sécurité. Les **accès sont restreints** dans pgAdmin et Firebase, et des **sauvegardes** PostgreSQL assurent la récupération en cas de problème.



(Voir Annexe 7 : Pare ChangePasswordFormType - Page 57)

- **Authentification et Formulaire** : Les formulaires sont protégés contre les attaques **CSRF** et **XSS** grâce à l'utilisation de **tokens CSRF** et du moteur de templates **Twig**. Cela empêche la soumission de formulaires frauduleux ou malveillants. **(Voir Annexe 6, 7 : Page 52 - 57)**

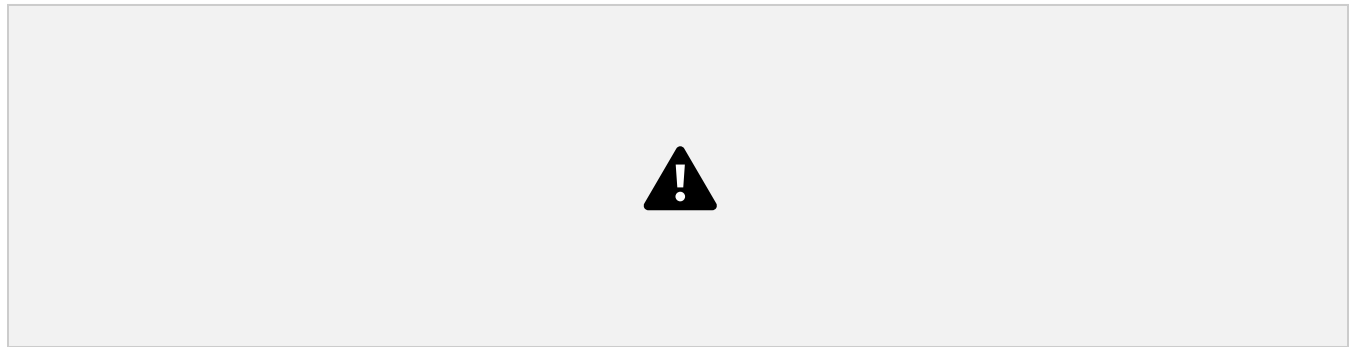
De plus, les sessions et cookies sont configurés de manière sécurisée avec Symfony, incluant les attributs **HttpOnly**, **Secure** et **SameSite=Lax**, pour protéger la confidentialité et

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

37

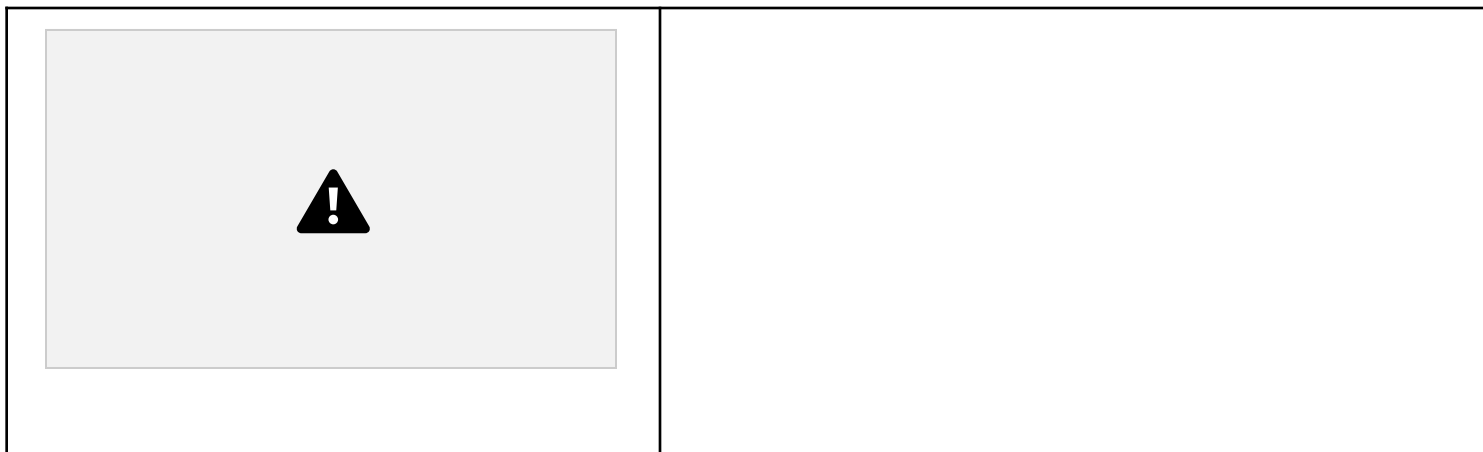
la sécurité des utilisateurs.





- **Rate Liming**

Pour prévenir les aaques par force brute, une fonconnalité de Rate Liming a été mise en place. Cela limite le nombre de tentaves de connexion qu'un utilisateur peut effectuer dans un laps de temps donné, protégeant ainsi le système contre les tentaves d'accès non autorisées.



- **Comptes Administrateurs**

Les comptes administrateurs sont créés via une commande sécurisée (php bin/console app:create-admin), avec une validaon rigoureuse des emails et des mots de passe, et prévenon des doublons. Les mots de passe sont masqués lors de la saisie, et l'ajout est confirmé dans pgAdmin. (**Voir Annexe 15 - Page 63**)



- **Ressources Frontend : Webpack Encore** minifie et regroupe les fichiers CSS et JS, améliorant ainsi la sécurité et la performance. Le **content hashing** garantit l'intégrité des fichiers, tandis que le **tree shaking** élimine le code inutilisé, réduisant ainsi la surface d'attaque et les risques.
(Voir Annexe 10 : Configuration Webpack - Page 59)
- **Environnements Sécurisés** : Docker, Symfony et GitHub Actions protègent les **variables sensibles** et **.gitignore** exclut les fichiers critiques.
- **Des messages de réussite et d'erreur**: Les messages sont affichés de manière sécurisée pour éviter toute fuite d'informations sensibles.

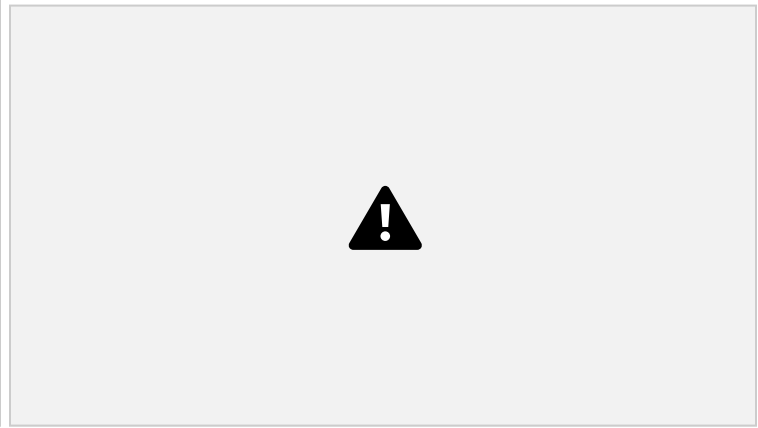
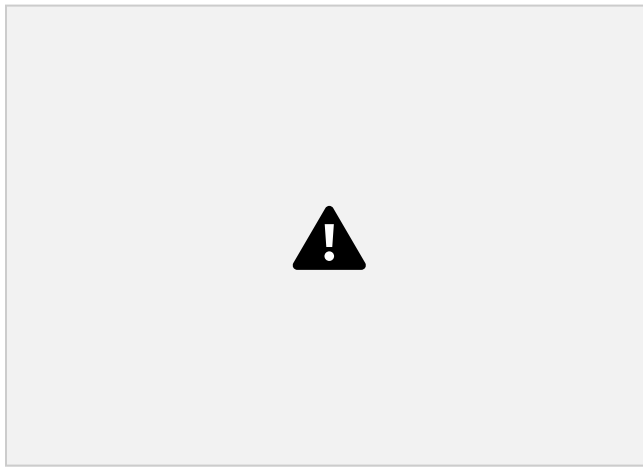
10. Gestion du Déploiement et Workflow CI/CD

Les étapes décrites ci-dessus représentent les bases essentielles du processus de déploiement. J'ai déjà présenté les différentes étapes du déploiement. À ce stade, je vais uniquement ajouter des précisions supplémentaires.

J'utilise Docker Compose pour déployer des services dans différents environnements à l'aide de fichiers dédiés (docker-compose.yml, .test.yml, .prod.yml). Les conteneurs sont lancés isolément avec **docker-compose -f [nom] up -d**, garantissant la cohérence des environnements.

(Voir Annexe 16, 18 - Page 64, 65)

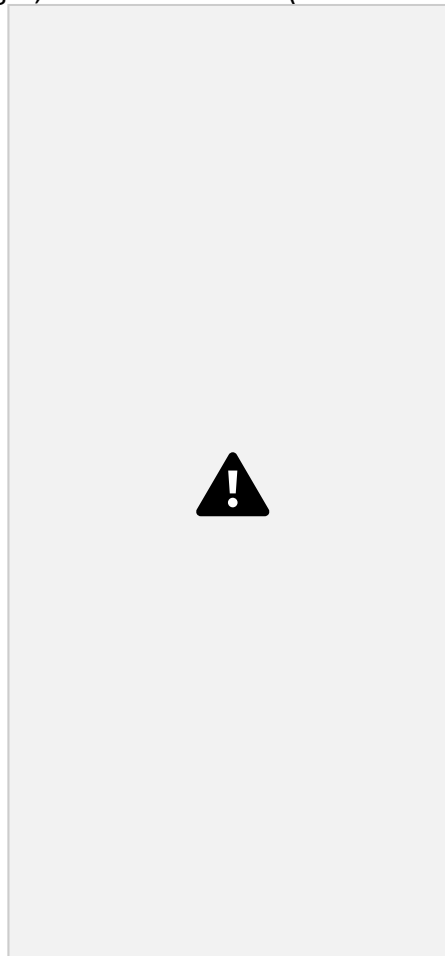
Les variables d'environnement et tokens, intégrés dans **le workflow GitHub Actions**, protègent les accès sensibles (PostgreSQL, clés GHCR). Le token GHCR facilite la gestion des images



Génération de Token GitHub Acons Mon pipeline CI/CD, structuré en deux étapes:

- **Build (Stage 1)** : Construon des images Docker à parr des sources et du Dockerfile puis envoi de celles-ci au registre GHCR.
- **Test (Stage 2)** : Tirage des images, exécution des tests (base de données, tests PHP, migraons) et validaon de leur stabilité.

Construon des images Docker Tirage des images et



tests

Résultat CI : Build et Test



exécution des

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

Pour les phases de **test** et de **producon**, j'utilise les mêmes images Docker, garantissant la cohérence et la fiabilité des déploiements. (**Voir Annexe 16, 18 - Page 64, 65**)

Pull d'image pour Docker Compose de producon



**Lancement des Services avec Docker Compose
pour l'environnement de Producon**



Docker-compose.prod.yml sur Docker desktop

- ❖ **Monitoring:** J'ai intégré **Prometheus** et **Grafana** pour surveiller les services. Des exportateurs spécifiques (PHP-FPM, Postgres, NGINX) collectent les métriques critiques, offrant une visibilité claire et rapide via des tableaux de bord Grafana. ([Voir Annexe 17 - Page 64](#))

Tableau de bord Grafana



Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh



11. Processus de recherche et traducon

Dans le cadre de mon projet, j'ai principalement consulté la documentaon Symfony et GitHub Docs ([Symfony Docs](#), [GitHub Docs](#)). Pour rechercher des informaons, j'ai privilégié les requêtes

en anglais sur Google, en utilisant des termes simples pour obtenir des explications claires et ai sélectionné les sources selon leur pertinence et actualité. J'ai aussi consulté des sites comme **Medium** et **Stack Overflow** pour compléter mes recherches.

❖ Ressource en anglais

Voici un extrait que j'ai choisi de traduire, provenant de la section "Securing Controllers and other Code" de la documentation. [Security \(Symfony Docs\)](#)

	<h4>❖ Traduction effectuée</h4> 
---	---

Conclusion

Qu'est ce que ce projet m'a appris ?

Ce projet m'a permis d'acquérir des compétences solides en développement d'application web avec Symfony, notamment la conception de bases de données, la création d'entités et de contrôleurs, ainsi que la gestion des routes et des vues. J'ai également maîtrisé l'utilisation de Docker et Docker Compose pour gérer des environnements de développement et de production cohérents, et intégré GitHub Actions pour automatiser les tests et les déploiements via un pipeline CI/CD.

Cette expérience m'a renforcé dans la gestion des données, la sécurité des applications et le déploiement, tout en me donnant une meilleure compréhension du fonctionnement global d'une

appliçon Symfony.

Les points à améliorer ?

Toutefois, certains aspects peuvent encore être améliorés, notamment le design frontend pour le rendre plus aractif et intuitif pour l'utilisateur.

Je pourrais également améliorer la couverture des tests avec PHPUnit et optimiser davantage la performance de l'appliçon. De plus, l'expérience utilisateur pourrait être affinée pour rendre l'appliçon encore plus réactive et agréable.

Enfin, bien que les mesures de sécurité de base soient en place, des renforcements supplémentaires seraient envisageables.

Qu'est ce que je vais faire plus tard ?

À court terme, je souhaite intégrer un poste de développeur back-end afin d'appliquer les compétences acquises durant ma formation et de me perfectionner sur le terrain.

À plus long terme, mon ambition est de me spécialiser davantage dans le développement back-end et l'ingénierie des données, tout en poursuivant mes études pour approfondir mes connaissances et relever des défis techniques plus complexes.

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

43

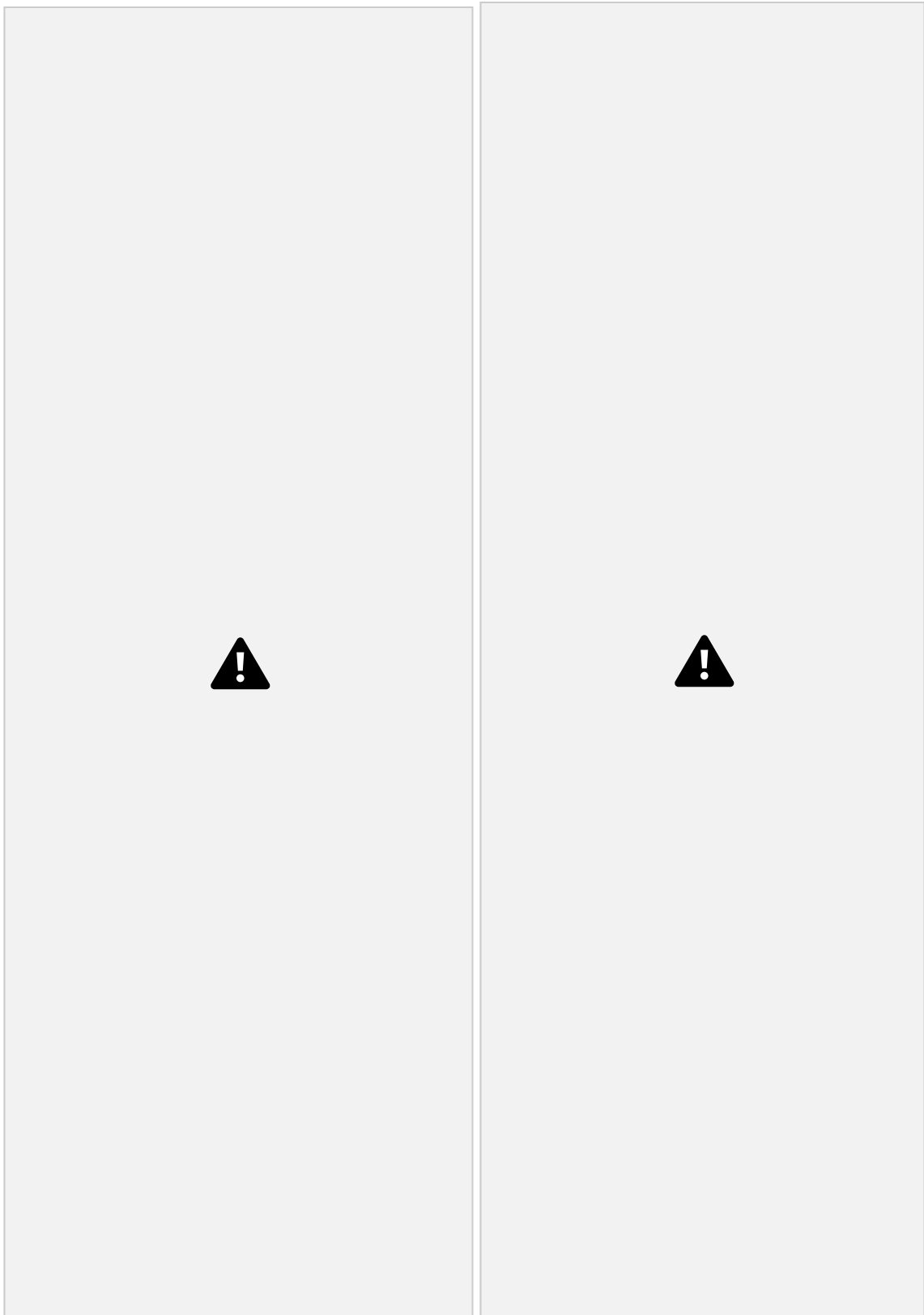
Annexes

Annexe 1 : Maquettes

Zoning - Accueil , Animal et Contact (desktop et mobile)



Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh



Wireframes et Prototypes - Accueil et Animal (desktop et mobile)

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

Wireframes - Contact et Connexion (desktop et mobile)



Prototypes - Contact et Connexion (desktop et mobile)



Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

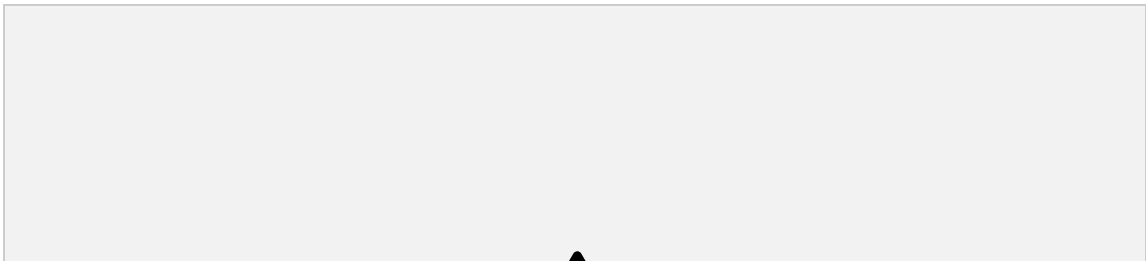
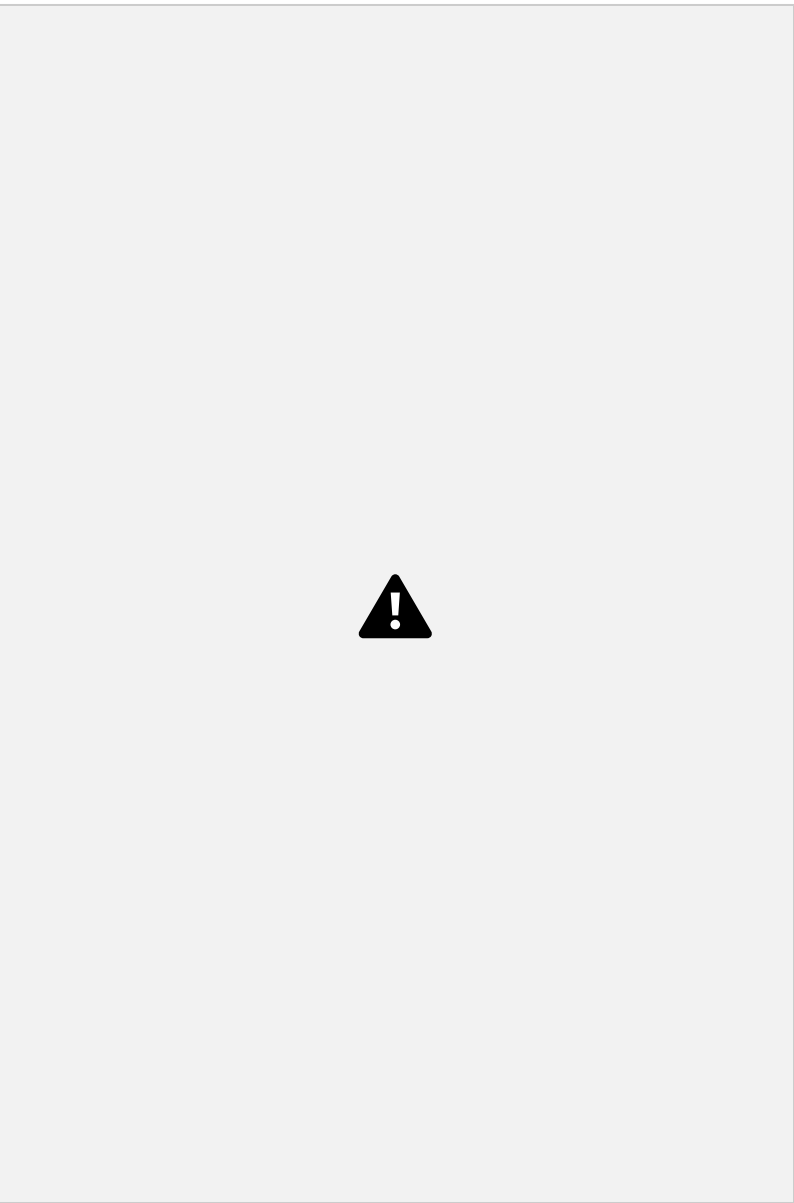
Wireframes - Tableau de bord et Vue Utilisateurs (desktop et mobile)

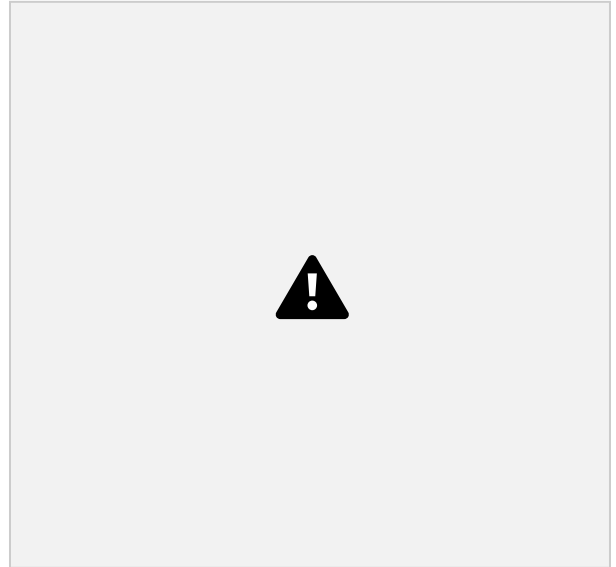


Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

Prototypes - Tableau de bord et Vue Utilisateurs (desktop et mobile)



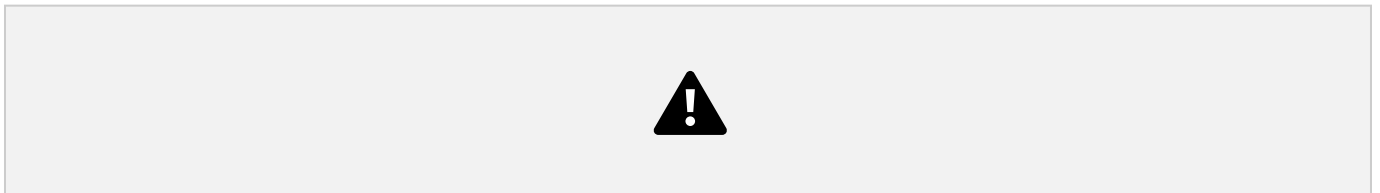


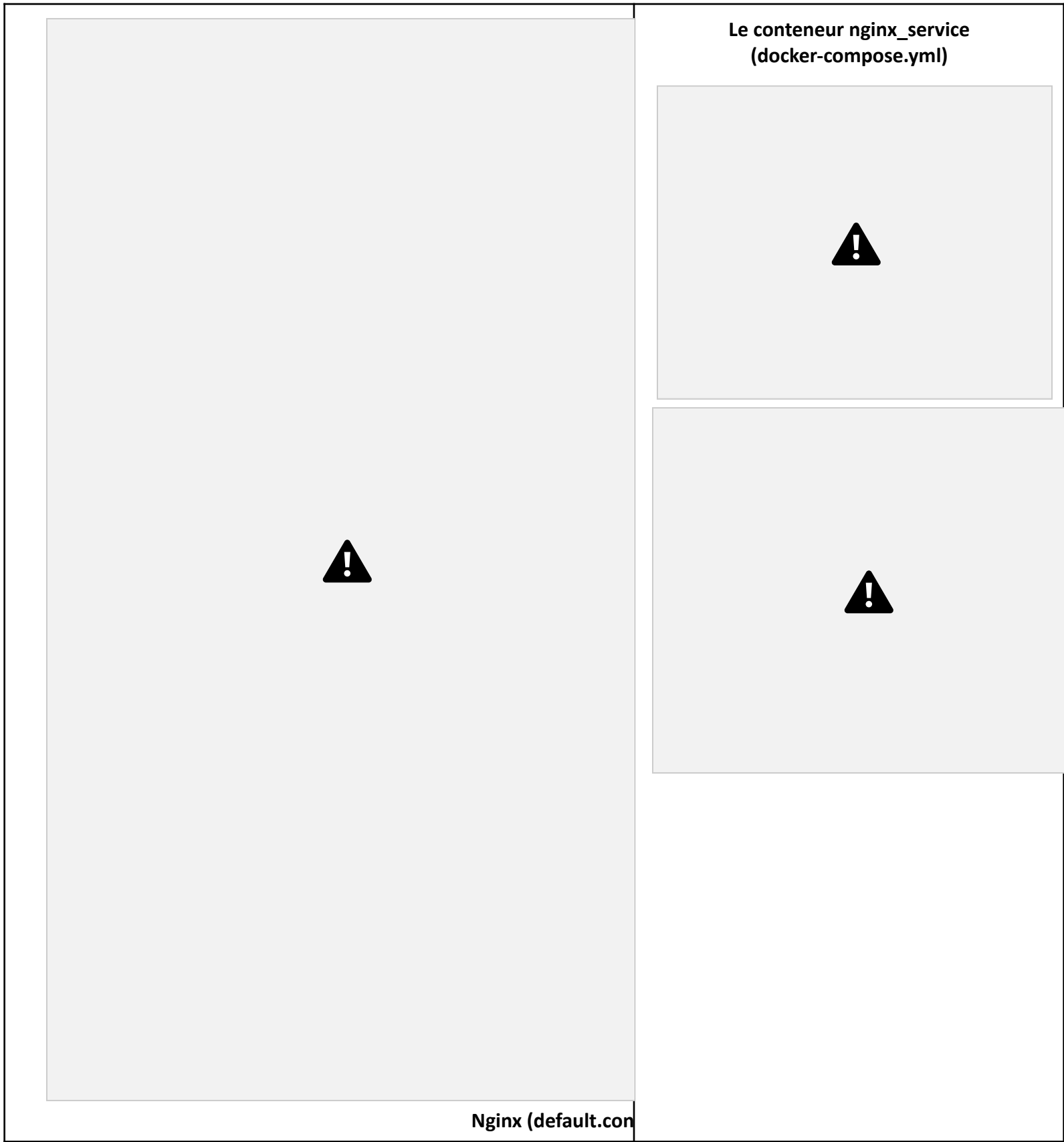


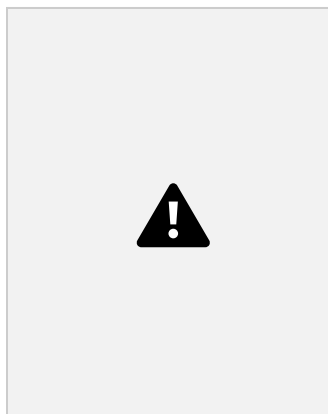
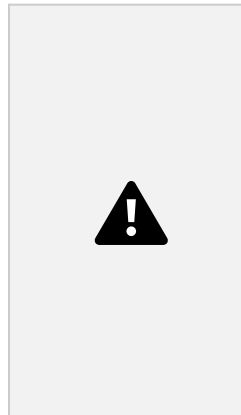
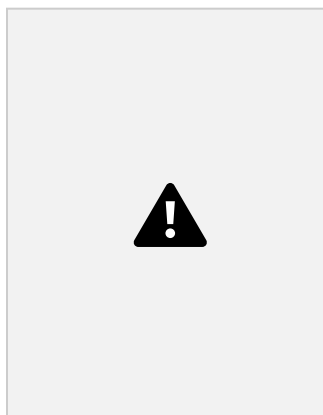
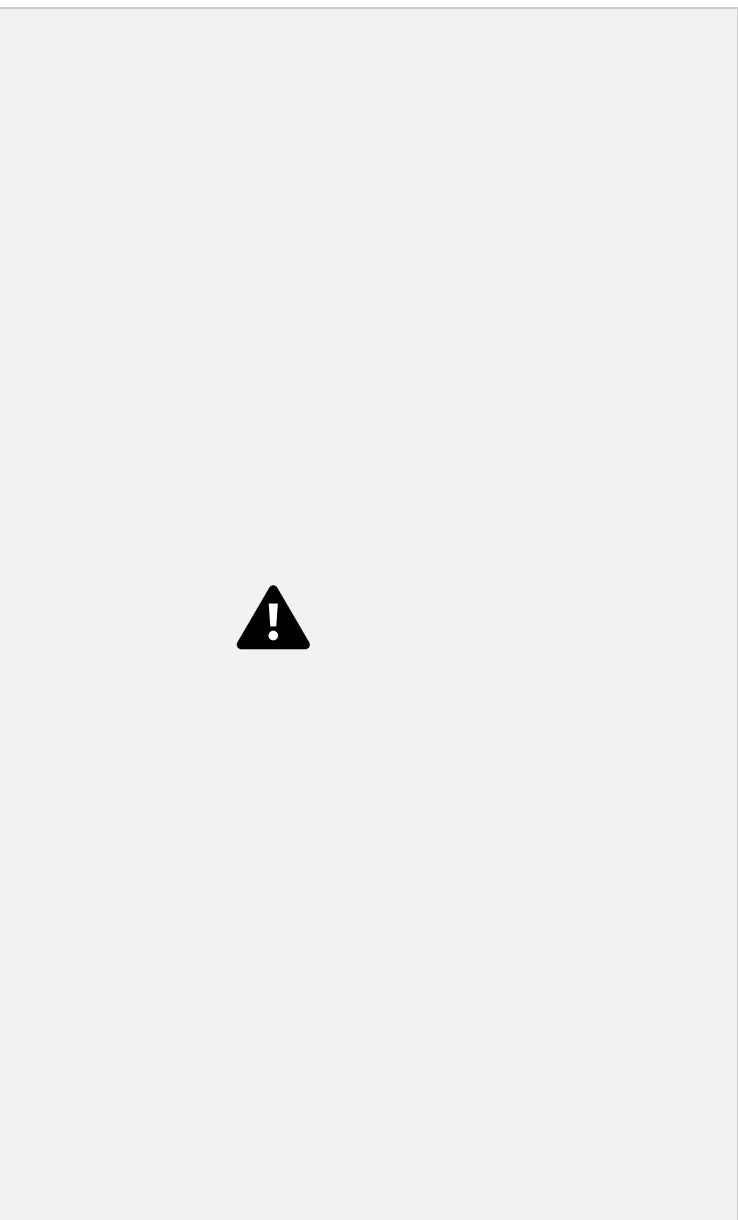
Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

Annexe 4 : Configuraon HTTPS

Un cerficat SSL auto-signé a été généré avec OpenSSL








MVC - Exemple Page Contact

- Page Contact Twig (Template) - Enté Contact



Annexe 6 : Système d’authenficaon (Login)

Généra	Contrôleur Authenficaon
<div data-bbox="112 1033 599 1917"></div> <div data-bbox="261 1929 602 1961">on de la structure de base</div>	

- Sécurité et gestion de l'authentification

- Twig Authenticaon





ResetPasswordController

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

56

- ChangePasswordFormType

Annexe 8 : Accès aux données NoSQL avec Firebase

- Exemple



de

code pour l'incrémentaon des vues dans Firebase via PHP - Exemple de code

de téléversement des images dans Firebase

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

57

Annexe 9 : CRUD ADMIN avec EasyAdmin

- Interface Vétérinaire







Annexe 11 : Optimisation SEO

- Optimisation des balises META



- Optimiser les images





Annexe 12 : Tests Unitaires avec PHPUnit

- Inialisaon des Tests avec PHPUnit
- Configuraon de l'Environnement de Test



- Exécuon des Migraons en environnement de Test



Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

60

- Généraon de Tests avec Symfony



- Sécurité Controller Test



Annexe 13 : Tests Fonconnels



- Saisie d'un formulaire de contact par l'utilisateur - Récepon du contact dans Maildev

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

61

- **Contact reçu dans le tableau de bord administrateur (Statut : Pending)**

- Liste des Utilisateurs

- Détails de l'utilisateur

- Table User dans PgAdmin



Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh

62

- Détails d'avis dans le tableau de bord

Menons Légale



**Annexe 15 : Créaon de
Compte Administrateur
avec Sécurité**





Annexe 16 : Docker-compose.test.yml

Annexe 17 : Configuraon de Prometheus (prometheus.yml)



[Annexe 18 : Docker-compose.prod.yml](#)

Dossier Projet - Zoo Arcadia - NGUYEN Thi Van Anh