SELLIER Eli

Documentation Technique du Projet ECF – Zoo Arcadia

Introduction

L'objectif de ce projet est de réaliser une application web robuste, répondant à des caractéristiques telles qu'une orientation mobile, un design adaptatif, et une facilité d'utilisation et d'entretien. J'ai choisi à ce titre un ensemble de technologies qui me permettent d'atteindre ces objectifs, définis dans le cahier des charges du projet.

Technologies choisies

Stack Front-End

HTML & CSS
Le **HTML** (Hyper-Text Markup Language) est le langage de base pour la structure de tout site web. Il organise l'aspect sémantique du contenu et joue un rôle crucial dans le référencement et l'accessibilité de l'application.
Le **CSS** (Cascading Style Sheets) est associé au HTML pour la mise en forme. Il permet de séparer le design du contenu, ce qui facilite la maintenance et les mises à jour.

Bootstrap avec SASS
Bootstrap est un framework CSS contenant une large variété de composants pré-construits. Il est facilement personnalisable, ce qui le rend adaptable à n'importe quel projet web. Ses principales qualités sont sa grille réactive et sa légèreté.
SASS (Syntactically Awesome Style Sheets) est un préprocesseur CSS qui étend les fonctionnalités de base du CSS avec des variables, des mixins et des fonctions. Cela permet d'automatiser des parties de code similaires, rendant le design plus facile à maintenir.

Stack Back-End

Symfony
Symfony est un framework PHP conçu pour produire des applications web de toutes tailles, organisé selon le modèle MVC (Model-View-Controller). Ses nombreux modules permettent de générer des éléments réutilisables, simplifiant considérablement le développement du back-end d'un site.

Justification des choix

HTML & CSS : Ces deux technologies constituent un standard de l'industrie depuis l'origine du Web. Elles sont largement supportées par tous les navigateurs actuels, garantissant une compatibilité universelle.

Bootstrap avec SASS : Le framework Bootstrap facilite le développement grâce à ses composants réutilisables, tandis que SASS se couple à CSS pour en optimiser l'emploi, évitant les redondances et centralisant des éléments, ce qui rend la maintenance plus simple et efficace.

Symfony : Symfony est un framework PHP reconnu pour sa robustesse et sa flexibilité. Il suit le modèle MVC (Model-View-Controller), permettant une séparation claire des préoccupations et une meilleure organisation du code. Parmi ses avantages, on trouve la modularité avec de nombreux bundles réutilisables, une communauté active et une documentation exhaustive, des fonctionnalités de sécurité intégrées, des performances optimales et une maintenance facilitée grâce à sa structure modulaire et sa large adoption.

Sécurité

Analyse des vulnérabilités :

- Pour garantir la sécurité de l'application, il est crucial d'identifier et de prévenir les vulnérabilités courantes telles que :
    - **Injection SQL** : Des attaques où des requêtes SQL malveillantes peuvent être insérées dans les entrées utilisateur pour manipuler la base de données.
    - **Cross-Site Scripting (XSS)** : Des attaques où des scripts malveillants sont injectés dans les pages web vues par d'autres utilisateurs.
    - **Cross-Site Request Forgery (CSRF)** : Des attaques où des actions non autorisées sont exécutées au nom de l'utilisateur authentifié.

Mesures de sécurité :

- **Authentification et gestion des rôles** : Utilisation de Symfony Security pour gérer les rôles et permissions des utilisateurs. Les rôles définis incluent les visiteurs, vétérinaires, employés et administrateurs.
- **Validation des entrées** : Utilisation des contraintes de validation de Symfony pour sécuriser les formulaires et s'assurer que toutes les entrées utilisateur sont correctement filtrées et validées.
- **Protection contre les attaques XSS et CSRF** :
    - Utilisation de la fonction `htmlspecialchars()` pour échapper les caractères spéciaux dans les entrées-utilisateur affichées.
    - Implémentation de jetons CSRF pour protéger les formulaires sensibles.

Configuration de l'environnement de travail

Installation des dépendances :

1. **Serveur WAMP** :
    a. Téléchargez et installez WAMP, qui comprend Apache, MySQL, et PHP, pour créer un serveur local sous Windows.
2. **Visual Studio Code** :
    a. Téléchargez et installez Visual Studio Code.
    b. Installez les extensions nécessaires pour le développement web et PHP, comme **PHP Intelephense**, **PHP Debug**, **Prettier** (pour le formatage du code), et **ESLint**.
3. **Symfony CLI** :
    a. Téléchargez l'installeur Symfony depuis Symfony et exécutez-le. Une fois installé, ouvrez une fenêtre de commande (cmd) et exécutez :

```
symfony
```

4. **Bootstrap via NPM** :
    a. Assurez-vous d'avoir Node.js installé. Téléchargez-le depuis Node.js.
    b. Dans une fenêtre de commande (cmd), exécutez :

```
npm install bootstrap@5.1.0
```

5. **SASS** :
    a. Dans une fenêtre de commande (cmd), exécutez :

```
npm install sass
```

Configuration des outils :

1. **Configurer WAMP** :
    a. Démarrez WAMP et assurez-vous que les services Apache et MySQL sont actifs.
2. **Configurer Symfony et Webpack Encore** :
    a. Créez un nouveau projet Symfony :

```
symfony new nom_de_projet --full
```

b. Configurez Webpack Encore pour gérer les assets :

```
const Encore = require('@symfony/webpack-encore');

Encore
  // Chemin du répertoire contenant vos fichiers
  .setOutputPath('public/build/')
  .setPublicPath('/build')

  // Ajout des entrées pour les assets
  .addEntry('app', './assets/app.js')

  // Activation de SASS loader
  .enableSassLoader()
  ...
```

3. **Connexion à GitHub depuis Visual Studio Code** :
   a. Ouvrez votre projet dans Visual Studio Code.
   b. Utilisez l'extension **GitHub** intégrée pour cloner votre dépôt :

```
Ctrl+Shift+P > Git: Clone
```

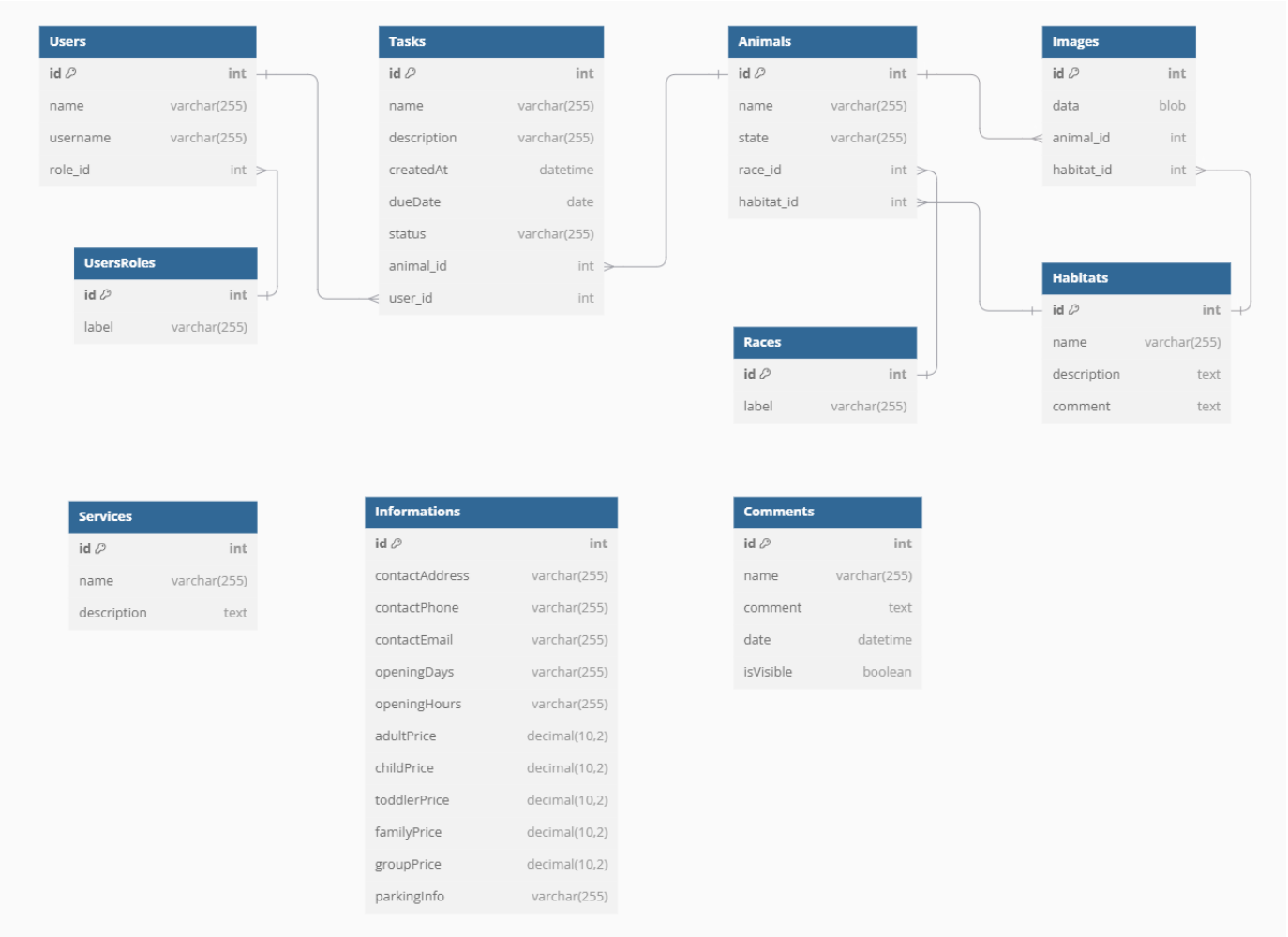
c. Suivez les instructions pour vous authentifier et cloner le dépôt.

Extensions recommandées pour Visual Studio Code :

- **PHP Intelephense** : Pour l'intelligence de code PHP.
- **PHP Debug** : Pour déboguer les scripts PHP.
- **Prettier - Code formatter** : Pour formater le code.
- **ESLint** : Pour identifier et corriger les problèmes dans le code JavaScript.

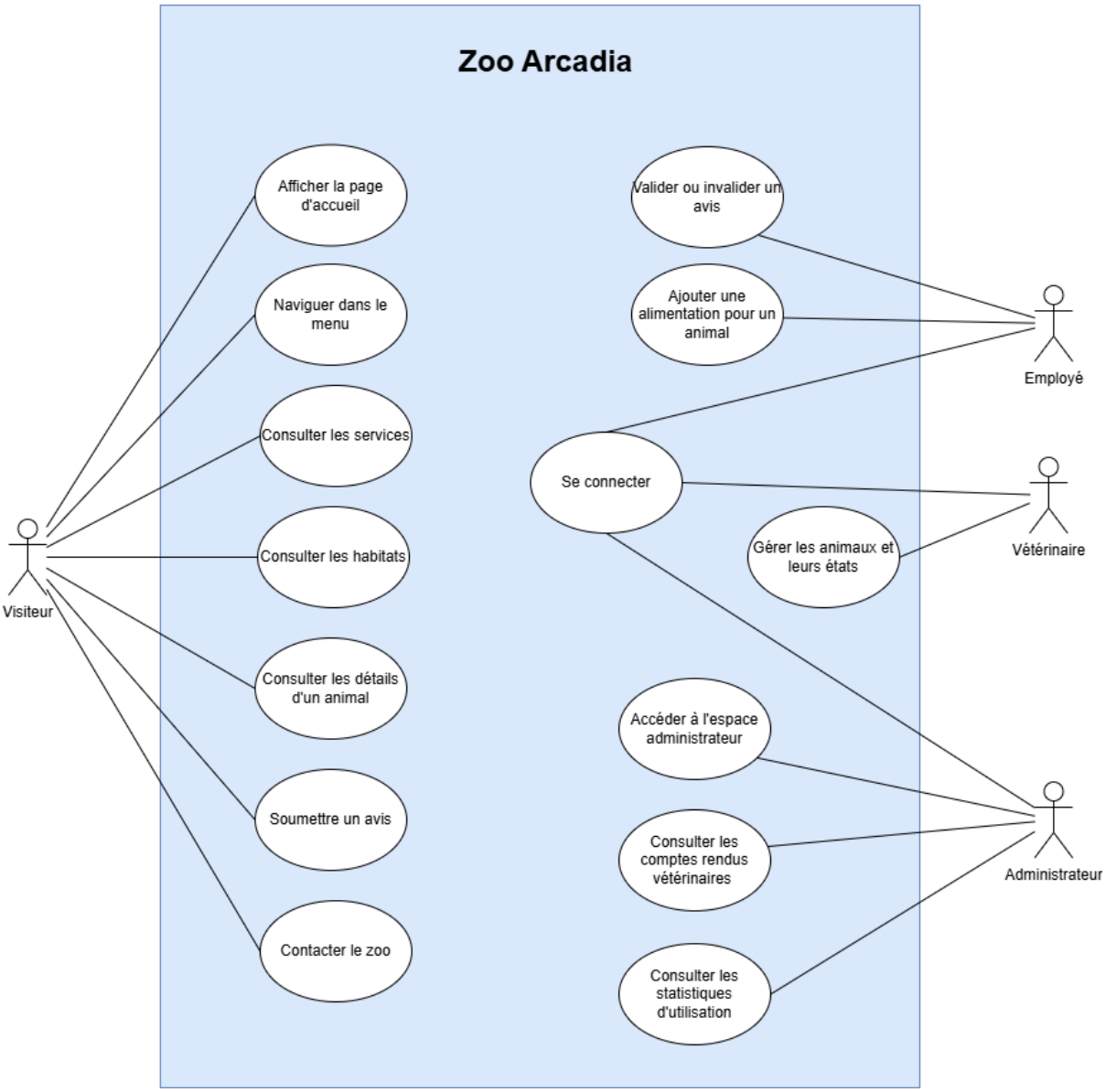
Modèle conceptuel de données

Le modèle conceptuel de données (MCD) pour le projet Zoo Arcadia comprend les entités suivantes et leurs relations :



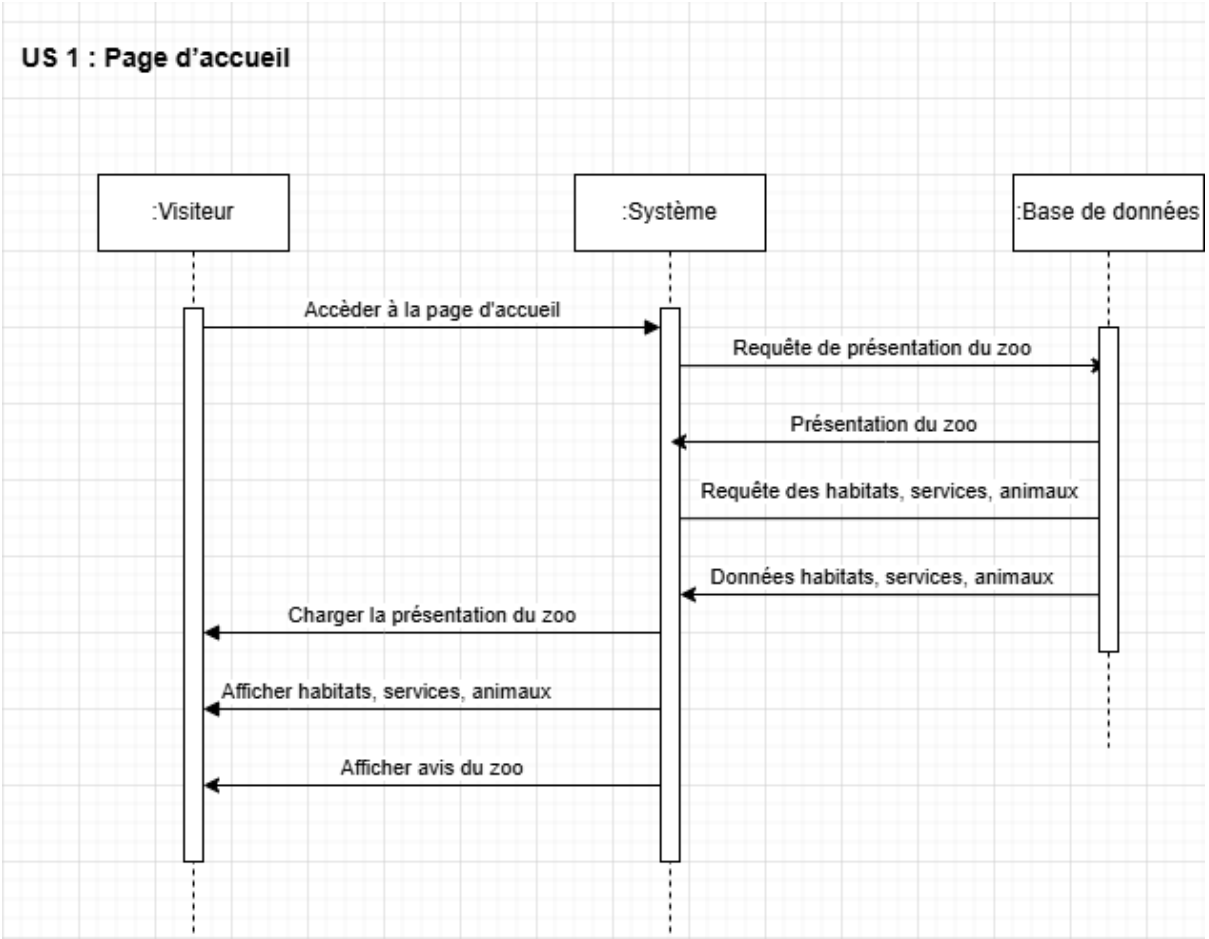
Diagrammes d'utilisation et de séquence

Diagrammes d'utilisation (Use Case) :

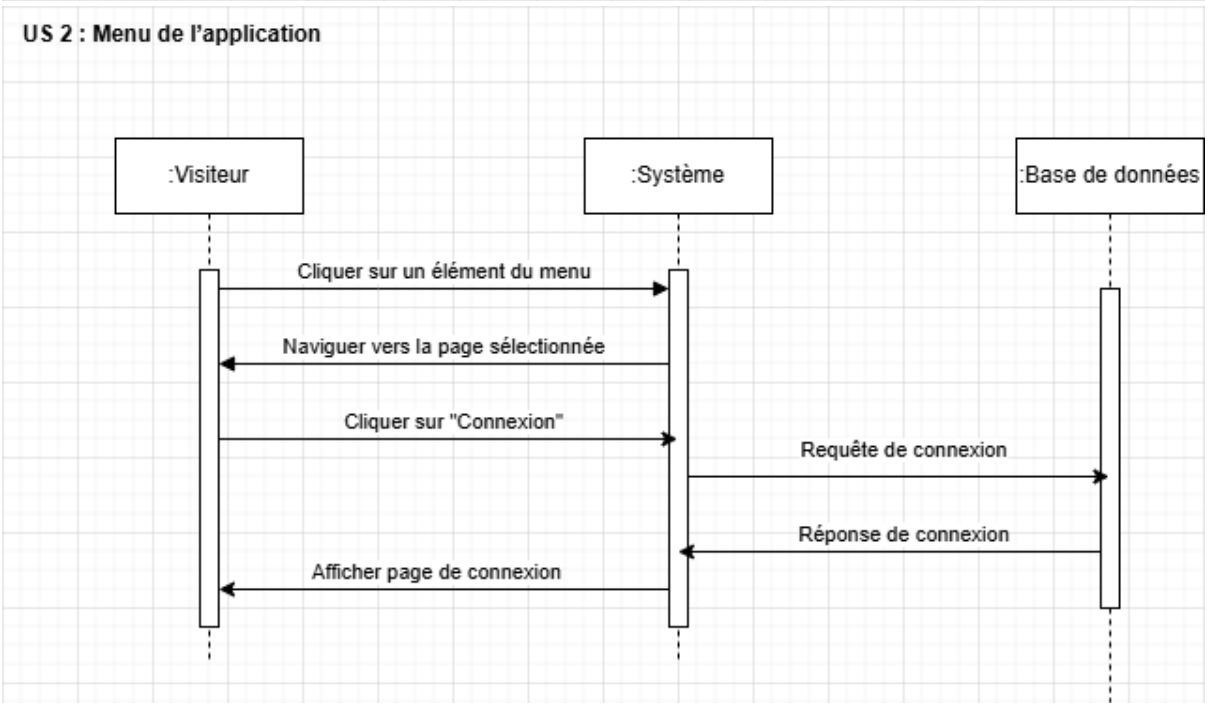


Diagrammes de séquence :

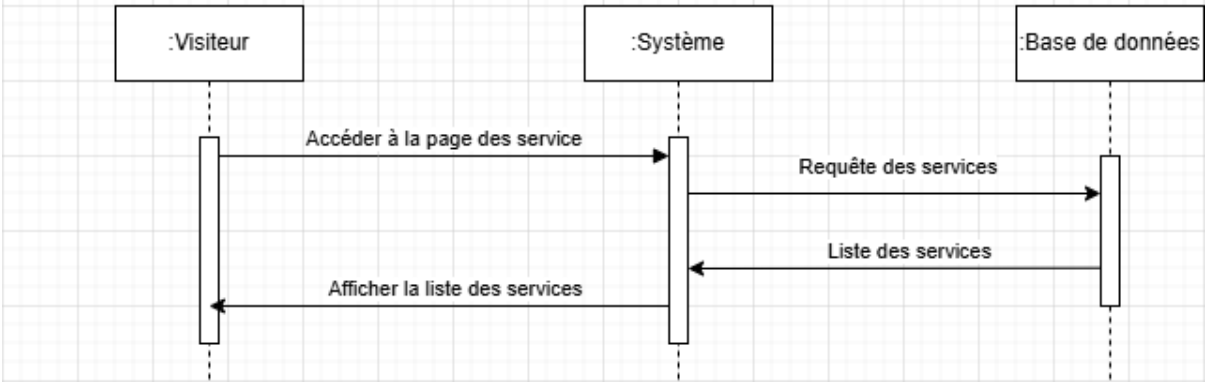
US 1 : Page d'accueil



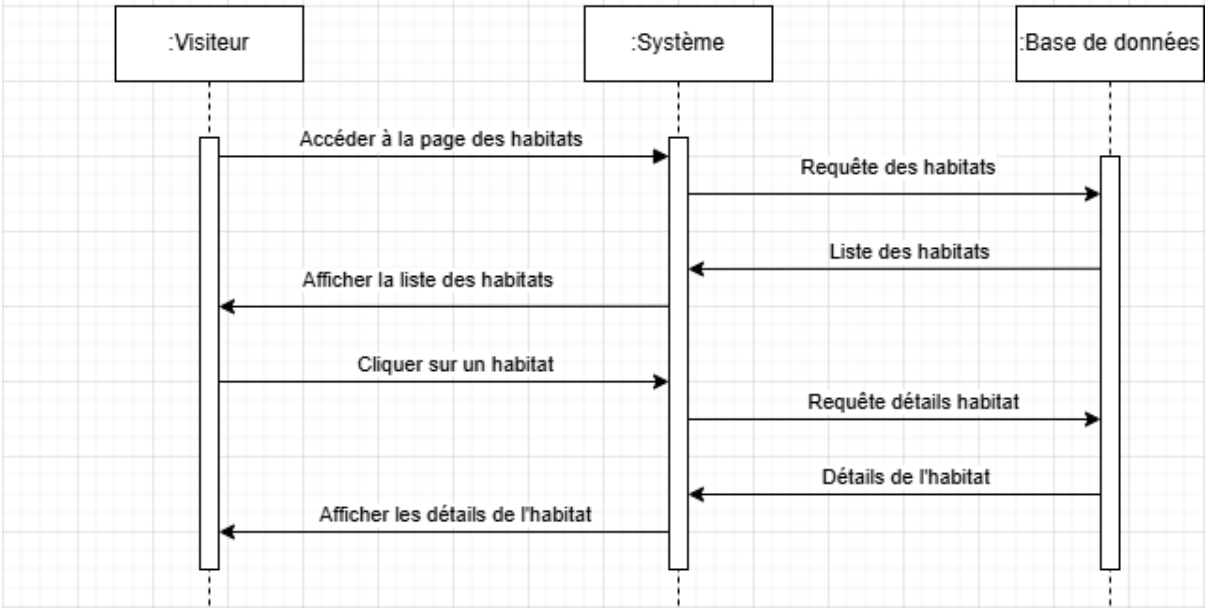
US 2 : Menu de l'application



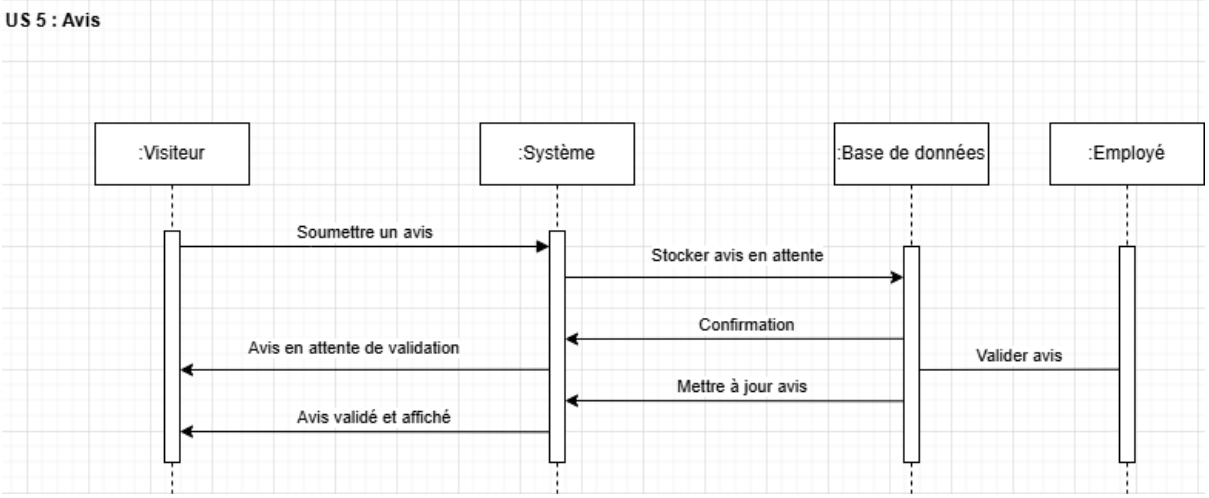
US 3 : Vue globale de tous les services



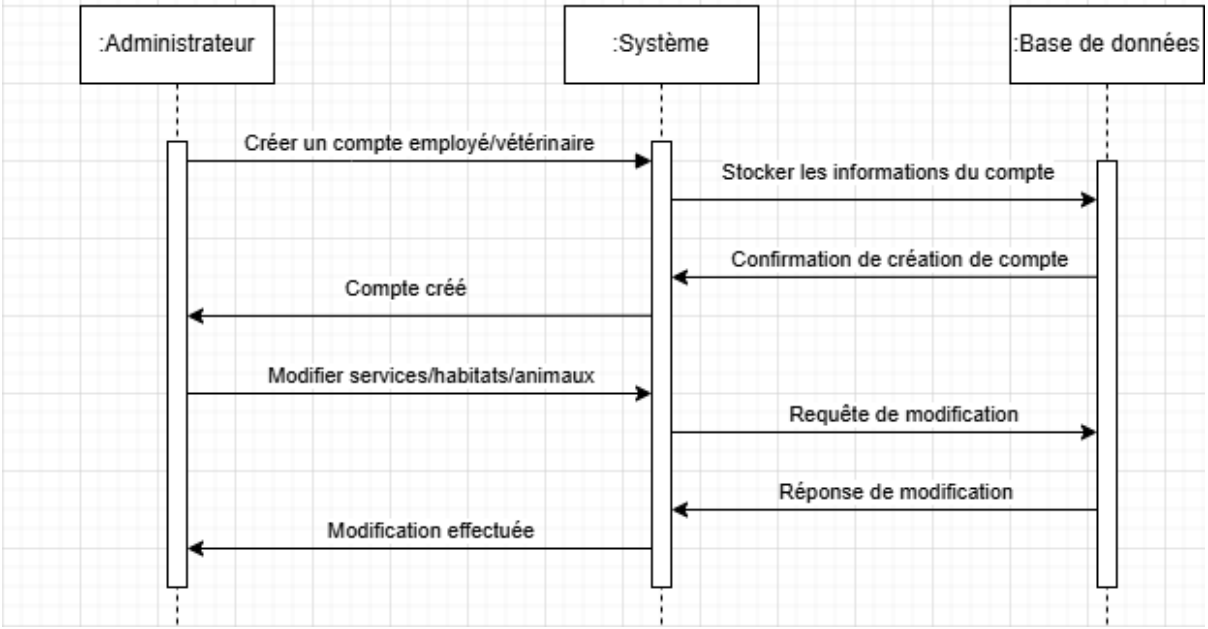
US 4 : Vue globale des habitats



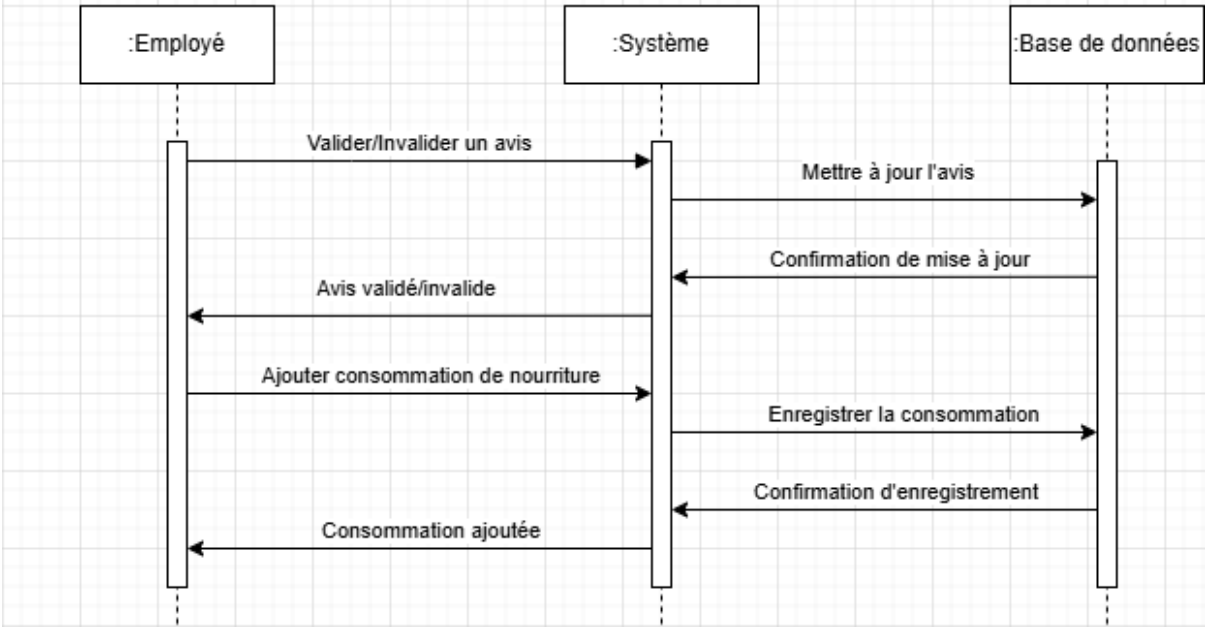
US 5 : Avis



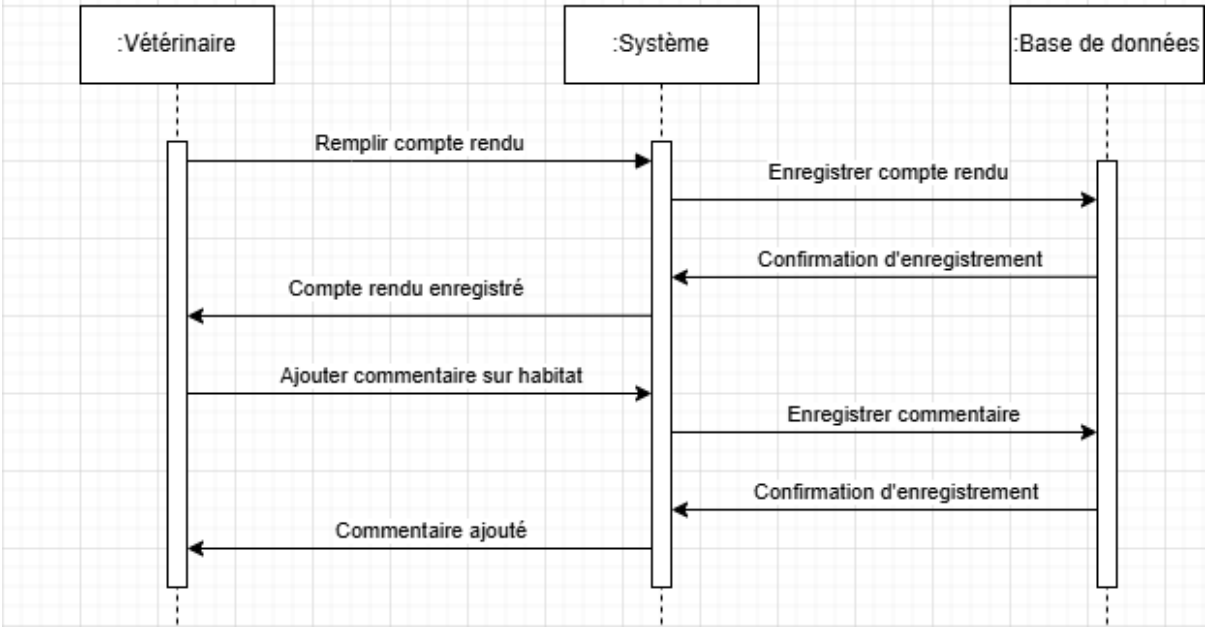
US 6 : Espace Administrateur



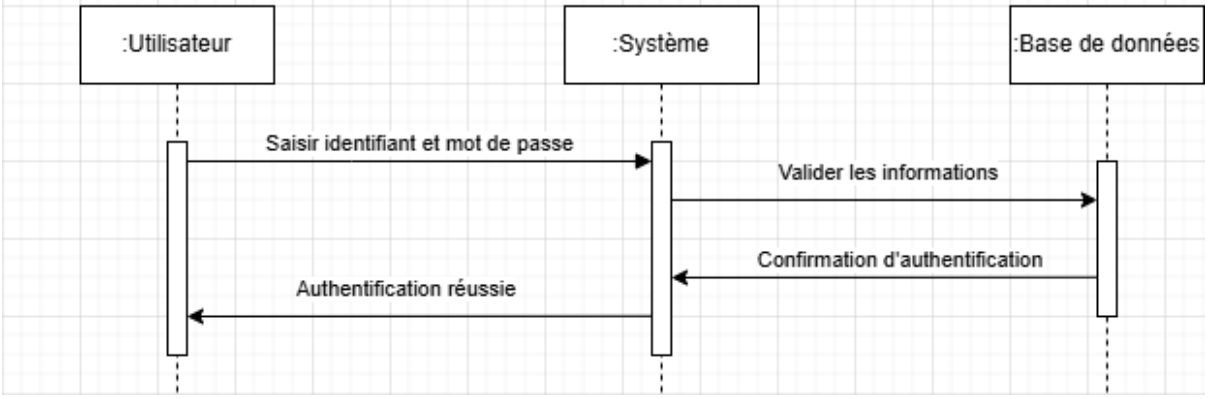
US 7 : Espace Employé

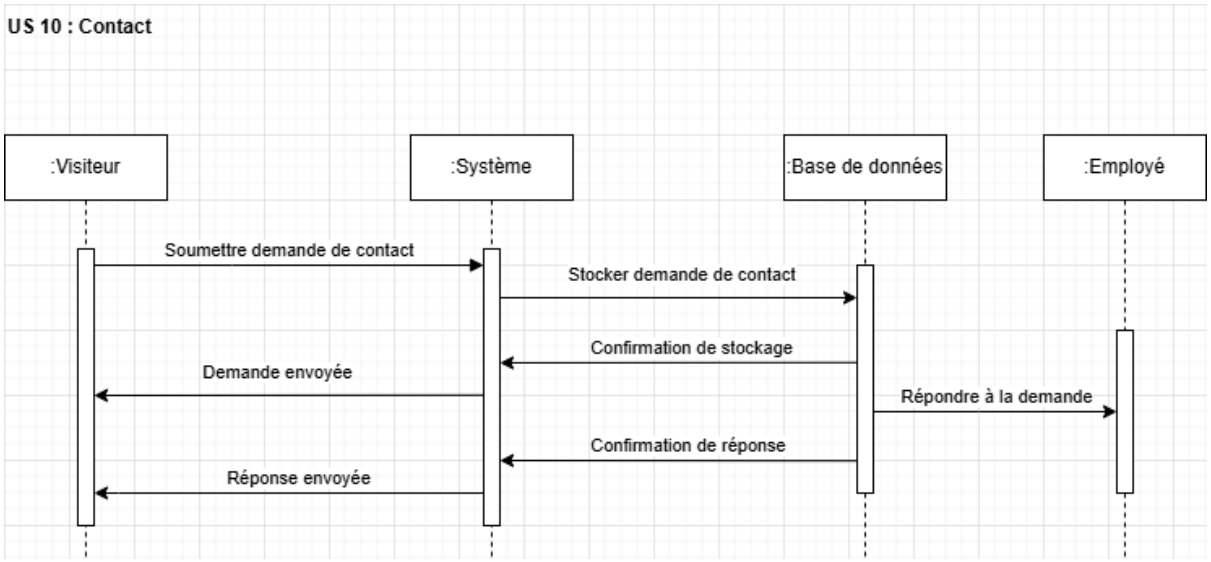


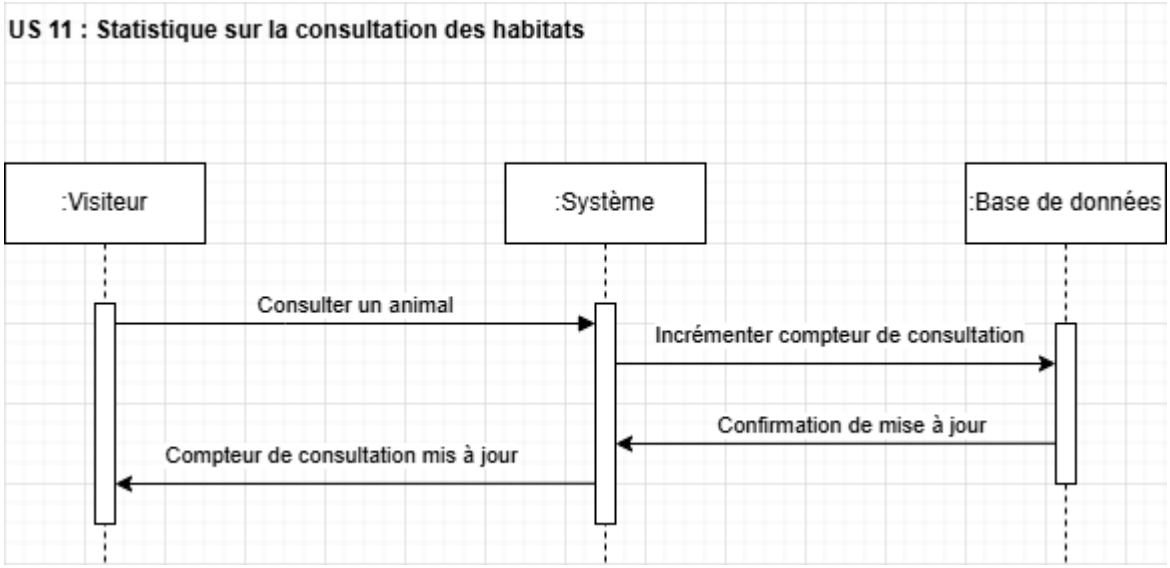
US 8 : Espace Vétérinaire



US 9 : Connexion







Déploiement de l'application

Instructions de déploiement :
Le déploiement de l'application se fait en plusieurs étapes pour s'assurer que tout est correctement configuré et fonctionnel.

1. **Préparation de l'environnement de production** :

    a. Installer les dépendances nécessaires.
    b. Configurer les variables d'environnement (ex. les informations de connexion à la base de données).

2. **Déploiement sur Heroku** (exemple de plateforme) :

    c. Initialiser un dépôt Git et créer une nouvelle application sur Heroku :

```
git init
heroku create nom_de_votre_application
git add .
git commit -m "Initial commit"
git push heroku main
```

d. Configurer les informations de connexion de la base de données sur Heroku :

```
heroku config:set
DATABASE_URL=mysql://user:password@hostname:port/dbname
```

3. Migrations de la base de données :

```
php bin/console doctrine:migrations:migrate
```

4. Créer les utilisateurs

php bin/console app:create-admin-user

5. Configuration du serveur Apache :

```
<VirtualHost *:80>
    ServerName zoo-arcadia.local
    DocumentRoot "C:/wamp64/www/zoo-arcadia/public"

    <Directory "C:/wamp64/www/zoo-arcadia/public">
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```