**Réflexions initiales technologiques sur le sujet :**  
Lorsque j'ai commencé à planifier le développement de notre application, j'ai pris en compte plusieurs aspects technologiques pour répondre aux besoins de notre projet. Voici quelques-unes des réflexions initiales que j'ai eues :

1. **Choix du langage de programmation :**
   * J'ai envisagé d'utiliser PHP comme langage principal pour le développement de l'application en raison de sa popularité pour les applications web et de sa compatibilité avec les serveurs web couramment utilisés comme Apache et Nginx.
   * PHP offre une grande flexibilité pour la création de sites web dynamiques et d'applications basées sur des bases de données.
2. **Gestion de la base de données :**
   * Pour stocker les données de l'application, j'ai opté pour MySQL, une solution de base de données relationnelle largement utilisée et bien documentée.
   * MySQL est réputé pour sa performance, sa fiabilité et sa compatibilité avec PHP, ce qui en fait un choix idéal pour notre application.
3. **Architecture de l'application :**
   * J'ai envisagé d'adopter une architecture MVC (Modèle-Vue-Contrôleur) pour organiser le code de l'application de manière modulaire et faciliter sa maintenance et son évolutivité.
   * Cette architecture permet de séparer la logique métier (Modèle), la présentation des données (Vue) et le contrôle des interactions utilisateur (Contrôleur).
4. **Sécurité des données :**
   * La sécurité des données est une préoccupation majeure, j'ai donc prévu d'implémenter des mesures de sécurité telles que la validation des données utilisateur, la protection contre les attaques CSRF (Cross-Site Request Forgery) et l'utilisation de requêtes préparées pour éviter les injections SQL.
5. **Expérience utilisateur :**
   * J'ai cherché à offrir une expérience utilisateur fluide et conviviale en concevant des interfaces utilisateur intuitives et en assurant des temps de chargement rapides des pages.

En résumé, ces réflexions initiales ont guidé mes choix technologiques pour développer une application web robuste, sécurisée et conviviale répondant aux besoins de notre projet.

**Documentation Technique de l'Application Web ZooArcadia Studio**

Cette documentation technique vise à fournir des instructions détaillées pour configurer l'environnement de travail et déployer l'application web ZooArcadia Studio localement. L'application est développée en PHP et utilise une base de données MySQL. Les outils et technologies principaux utilisés sont XAMPP, Visual Studio Code, Git/GitHub et PHPMyAdmin.

**Configuration de l'Environnement de Travail**

**1. Installation de XAMPP**

* Téléchargez XAMPP à partir du site officiel : <https://www.apachefriends.org/index.html>
* Suivez les instructions d'installation pour votre système d'exploitation.

**2. Configuration de XAMPP**

* Lancez XAMPP.
* Démarrez les services Apache et MySQL en cliquant sur les boutons "Start" correspondants dans la fenêtre de contrôle de XAMPP.

**3. Installation de Visual Studio Code**

* Téléchargez et installez Visual Studio Code depuis le site officiel : <https://code.visualstudio.com/>
* Suivez les instructions d'installation pour votre système d'exploitation.

**4. Installation de Git et Configuration de GitHub**

* Téléchargez et installez Git depuis le site officiel : <https://git-scm.com/downloads>
* Créez un compte sur GitHub : <https://github.com/>
* Configurez Git avec votre nom d'utilisateur et votre adresse e-mail.

**5. Installation de PHPMyAdmin**

* PHPMyAdmin est inclus dans XAMPP. Assurez-vous que les services Apache et MySQL sont en cours d'exécution, puis accédez à <http://localhost/phpmyadmin/> dans votre navigateur pour l'utiliser.

**Déploiement de l'Application Web**

**1. Téléchargement du Code Source**

* Clonez le dépôt GitHub de l'application ZooArcadia Studio en utilisant la commande suivante dans votre terminal :

‘git clone https://github.com/STARK2612/STUDI.git’

**2. Placement du Code Source dans le Répertoire htdocs de XAMPP**

* Naviguez jusqu'au répertoire d'installation de XAMPP sur votre système.
* Trouvez le répertoire "htdocs" et copiez les fichiers de votre application dans ce répertoire.

**3. Configuration de la Base de Données**

* Assurez-vous que MySQL est en cours d'exécution dans XAMPP.
* Accédez à <http://localhost/phpmyadmin/> dans votre navigateur.
* Créez une nouvelle base de données et importez le fichier SQL fourni avec l'application.

**4. Modification de la Configuration de l'Application**

* Renommez le fichier de configuration fourni (s'il en existe un, comme config.php) pour qu'il corresponde à vos paramètres locaux.
* Assurez-vous que les paramètres de connexion à la base de données correspondent à ceux de votre environnement local (nom d'utilisateur, mot de passe, nom de la base de données, etc.).

**5. Accès à l'Application Web**

* Ouvrez votre navigateur et accédez à <http://localhost/STUDI/index.php>.
* Remplacez "STUDI" par le nom du répertoire où vous avez placé votre application.

Conclusion

En suivant ces étapes, vous pourrez configurer et déployer l'application web ZooArcadia Studio localement sur votre machine. Ce processus utilise XAMPP comme serveur local, Visual Studio Code comme éditeur de code, Git/GitHub pour le contrôle de version, et PHPMyAdmin pour la gestion de la base de données.

**Modèle Conceptuel de Données (Diagramme de Classe) :**

Pour concevoir notre application, j'ai élaboré un modèle conceptuel de données afin de représenter les entités principales et leurs relations. Voici une description de ce modèle :

1. **Entités principales** :
   * **Utilisateur** : Cette entité représente les utilisateurs de l'application. Elle contient des informations telles que le nom d'utilisateur, le mot de passe haché, le nom, le prénom et le rôle de l'utilisateur.
   * **Role** : Cette entité représente les différents rôles attribués aux utilisateurs. Elle contient des informations telles que l'identifiant du rôle et son libellé.
2. **Relations** :
   * **Utilisateur - Role** : Il s'agit d'une relation entre l'entité Utilisateur et l'entité Role. Chaque utilisateur est associé à un rôle spécifique.
3. **Attributs** :
   * L'entité Utilisateur possède les attributs suivants :
     + username : le nom d'utilisateur unique.
     + password : le mot de passe haché pour l'authentification.
     + nom : le nom de l'utilisateur.
     + prenom : le prénom de l'utilisateur.
   * L'entité Role possède les attributs suivants :
     + role\_id : l'identifiant unique du rôle.
     + label : le libellé du rôle (par exemple : "Administrateur", "Employé", "Vétérinaire").

En utilisant ce modèle conceptuel de données, j'ai pu définir la structure de la base de données de notre application et mettre en place les relations nécessaires entre les entités. Cela nous permet d'avoir une vue claire de la façon dont les données sont organisées et interconnectées dans notre système.

**Diagramme d'Utilisation et Diagramme de Séquence :**

Pour comprendre le flux d'utilisation de notre application, j'ai élaboré un diagramme d'utilisation ainsi qu'un diagramme de séquence. Voici une description de ces diagrammes :

1. **Diagramme d'Utilisation** :
   * Ce diagramme présente les différentes actions ou fonctionnalités de l'application telles que la création de compte, la modification de compte et l'authentification.
   * Chaque action est représentée par un cas d'utilisation, qui décrit les interactions entre les acteurs (utilisateurs) et le système.
2. **Diagramme de Séquence** :
   * Ce diagramme illustre la séquence d'actions qui se produit lorsqu'un utilisateur interagit avec le système pour effectuer une tâche spécifique, comme la création ou la modification d'un compte utilisateur.
   * Il met en évidence les étapes chronologiques de l'interaction entre l'utilisateur et le système, en montrant les messages échangés entre les différents objets ou composants impliqués dans le processus.

Documentation du Déploiement :  
Pour déployer notre application, j'ai suivi une démarche méthodique en plusieurs étapes. Voici une explication détaillée de chaque étape :

**1.1 Création du Serveur Web avec Alwaysdata.com :**

* Rendez-vous sur le site d'Alwaysdata.com et connectez-vous à votre compte.
* Accédez à la section de gestion des serveurs web et créez un nouveau serveur web.
* Choisissez Apache ou Nginx comme serveur web en fonction de vos préférences ou des besoins spécifiques de votre application.
* Configurez les paramètres du serveur web en définissant le répertoire racine et en activant les modules nécessaires comme PHP si requis.

**1.2 Configuration du Système de Gestion de Base de Données avec Alwaysdata.com :**

* Sur Alwaysdata.com, accédez à la section de gestion des bases de données et créez une nouvelle base de données MySQL ou MariaDB.
* Spécifiez le nom de la base de données, l'utilisateur et le mot de passe.
* Assurez-vous de noter les informations de connexion à la base de données car elles seront nécessaires lors de la configuration de votre application.
* correctement interprété, et créez un fichier HTML pour vérifier si le serveur web fonctionne comme prévu.

1. **Acquisition du Code Source** :
   * Ensuite, j'ai obtenu le code source de l'application à partir du référentiel de versionnement (Git, SVN) ou en téléchargeant l'archive du code depuis une source externe.
   * J'ai vérifié que le code source était complet et ne comportait pas d'erreurs ou d'éléments manquants.
2. **Configuration de la Base de Données** :
   * J'ai créé une base de données sur mon serveur de base de données (MySQL, MariaDB) en utilisant les paramètres définis dans le code source de l'application.
   * J'ai importé les fichiers de structure de la base de données et éventuellement des données de test si nécessaire.
3. **Configuration du Serveur Web** :
   * J'ai configuré mon serveur web (Apache, Nginx) pour qu'il pointe vers le répertoire racine de l'application.
   * J'ai vérifié les autorisations d'accès aux fichiers et répertoires pour m'assurer que le serveur web pouvait lire et exécuter les fichiers de l'application.
4. **Tests de Déploiement** :
   * Avant de rendre l'application disponible au public, j'ai effectué des tests de déploiement pour vérifier son bon fonctionnement.
   * J'ai testé les fonctionnalités principales de l'application, y compris l'authentification, la création de comptes utilisateur et la modification de comptes.
5. **Déploiement sur le Serveur de Production** :
   * Une fois que j'ai confirmé que l'application fonctionnait correctement en environnement de développement, j'ai procédé au déploiement sur mon serveur de production.
   * J'ai transféré tous les fichiers de l'application sur le serveur de production et j'ai effectué les configurations nécessaires pour que l'application soit accessible via Internet.
6. **Surveillance et Maintenance** :
   * Enfin, j'ai mis en place des outils de surveillance pour surveiller les performances de l'application et détecter d'éventuels problèmes.
   * J'ai planifié des interventions de maintenance régulières pour appliquer les mises à jour de sécurité et corriger les éventuels bugs découverts après le déploiement.

En suivant ces étapes, j'ai réussi à déployer notre application de manière sécurisée et fonctionnelle, prête à être utilisée par les utilisateurs finaux.

Documentation du Déploiement :  
Pour déployer notre application, j'ai suivi une démarche méthodique en plusieurs étapes. Voici une explication détaillée de chaque étape :

1. **Configuration de l'Environnement de Développement** :
   * Tout d'abord, j'ai configuré mon environnement de développement en installant les outils nécessaires tels que le serveur web (Apache, Nginx), le système de gestion de base de données (MySQL, MariaDB), et PHP.
   * J'ai vérifié que tous les logiciels étaient correctement installés et configurés en fonction de mes besoins.
2. **Acquisition du Code Source** :
   * Ensuite, j'ai obtenu le code source de l'application à partir du référentiel de versionnement (Git, SVN) ou en téléchargeant l'archive du code depuis une source externe.
   * J'ai vérifié que le code source était complet et ne comportait pas d'erreurs ou d'éléments manquants.
3. **Configuration de la Base de Données** :
   * J'ai créé une base de données sur mon serveur de base de données (MySQL, MariaDB) en utilisant les paramètres définis dans le code source de l'application.
   * J'ai importé les fichiers de structure de la base de données et éventuellement des données de test si nécessaire.
4. **Configuration du Serveur Web** :
   * J'ai configuré mon serveur web (Apache, Nginx) pour qu'il pointe vers le répertoire racine de l'application.
   * J'ai vérifié les autorisations d'accès aux fichiers et répertoires pour m'assurer que le serveur web pouvait lire et exécuter les fichiers de l'application.
5. **Tests de Déploiement** :
   * Avant de rendre l'application disponible au public, j'ai effectué des tests de déploiement pour vérifier son bon fonctionnement.
   * J'ai testé les fonctionnalités principales de l'application, y compris l'authentification, la création de comptes utilisateur et la modification de comptes.
6. **Déploiement sur le Serveur de Production** :
   * Une fois que j'ai confirmé que l'application fonctionnait correctement en environnement de développement, j'ai procédé au déploiement sur mon serveur de production.
   * J'ai transféré tous les fichiers de l'application sur le serveur de production et j'ai effectué les configurations nécessaires pour que l'application soit accessible via Internet.
7. **Surveillance et Maintenance** :
   * Enfin, j'ai mis en place des outils de surveillance pour surveiller les performances de l'application et détecter d'éventuels problèmes.
   * J'ai planifié des interventions de maintenance régulières pour appliquer les mises à jour de sécurité et corriger les éventuels bugs découverts après le déploiement.

En suivant ces étapes, j'ai réussi à déployer notre application de manière sécurisée et fonctionnelle, prête à être utilisée par les utilisateurs finaux.

Dans la gestion de ce projet, j'ai suivi une approche méthodique pour assurer son succès. Voici une explication de ma gestion de projet :

1. **Définition des Objectifs** :
   * Tout d'abord, j'ai clairement défini les objectifs de l'application en tenant compte des besoins des utilisateurs et des exigences du projet.
2. **Planification du Projet** :
   * Ensuite, j'ai établi un plan de projet détaillé en identifiant les tâches à accomplir, les délais, les ressources nécessaires et les responsabilités de chacun.
3. **Organisation de l'Équipe** :
   * J'ai constitué une équipe compétente en fonction des besoins du projet, en affectant des rôles et des responsabilités clairs à chaque membre de l'équipe.
4. **Communication et Collaboration** :
   * J'ai encouragé une communication ouverte et transparente au sein de l'équipe, en organisant régulièrement des réunions de suivi pour discuter de l'avancement du projet et résoudre les problèmes éventuels.
5. **Gestion des Risques** :
   * J'ai identifié les risques potentiels liés au projet et mis en place des stratégies pour les atténuer ou les éviter autant que possible.
6. **Suivi et Contrôle** :
   * J'ai surveillé de près l'avancement du projet en utilisant des outils de suivi et de gestion de projet, en m'assurant que le projet respecte les délais et les budgets prévus.
7. **Tests et Validation** :
   * Avant la livraison finale, j'ai effectué des tests approfondis pour vérifier que l'application répondait aux exigences fonctionnelles et de qualité spécifiées.
8. **Formation et Support** :
   * J'ai fourni une formation adéquate aux utilisateurs finaux de l'application et mis en place un système de support pour répondre à leurs questions et résoudre leurs problèmes éventuels.
9. **Livraison et Déploiement** :
   * Enfin, j'ai assuré une livraison réussie de l'application en respectant les délais convenus et en assurant un déploiement fluide sur les serveurs de production.

En suivant cette approche de gestion de projet, j'ai pu mener le projet à bien et livrer une application de haute qualité qui répond aux besoins des utilisateurs.

Haut du formulaire

Pour la gestion de projet de notre application, j'ai opté pour l'utilisation d'un tableau Kanban partagé. Ce tableau nous permet d'avoir une vue d'ensemble de notre progression et de suivre l'avancement des fonctionnalités prévues. Voici comment je l'ai organisé :

1. **Colonnes des Fonctionnalités Prévues** :
   * Dans cette colonne, je recense toutes les fonctionnalités prévues pour l'application, classées par ordre de priorité. Cela nous aide à avoir une vision claire de ce qui doit être développé.
2. **Colonnes des Fonctionnalités en Cours de Développement** :
   * Dans cette colonne, je liste les fonctionnalités que je suis en train de développer ou celles prévues pour le sprint en cours si nous utilisons une méthodologie agile. Cela nous permet de savoir sur quoi je travaille actuellement.
3. **Colonnes des Fonctionnalités Terminées (sur la Branche de Développement)** :
   * Une fois qu'une fonctionnalité est développée et testée sur la branche de développement, je la déplace dans cette colonne. Cela indique que la fonctionnalité est prête à être mergée dans la branche principale.
4. **Colonnes des Fonctionnalités Mergées dans la Branche Principale** :
   * Enfin, dans cette colonne, je liste les fonctionnalités qui ont été mergées avec succès dans la branche principale du dépôt de code. Cela nous permet de suivre les fonctionnalités qui sont disponibles dans la version actuelle de l'application.

En utilisant ce tableau Kanban, je peux facilement suivre l'état d'avancement du projet, identifier les éventuels blocages et assurer une progression efficace vers la réalisation de toutes les fonctionnalités prévues.

+-----------------------------+-------------------------+---------------------------+-------------------------+-----------------------------------+

| Fonctionnalités Prévues | Fonctionnalités en Cours | Fonctionnalités Terminées | Fonctionnalités Mergées | Fonctionnalités Livrées |

+-----------------------------+-------------------------+---------------------------+-------------------------+-----------------------------------+

| 1. Ajout d'utilisateurs | 4. Gestion des rôles | 7. Interface admin | 10. Authentification | 13. Interface employé |

| 2. Modification de comptes | 5. Page de connexion | 8. Formulaire d'inscription| 11. Gestion des sessions| 14. Interface vétérinaire |

| 3. Suppression d'utilisateurs| 6. Page de profil | 9. Base de données | 12. Sécurité CSRF | 15. Rapports de statistiques |

+-----------------------------+-------------------------+---------------------------+-------------------------+-----------------------------------+  
  
  
  
  
| Fonctionnalités Prévues | En Développement | En Test | Terminées (Br. Dévelop.) | Mergées (Br. Principale) |

|--------------------------|------------------|---------|--------------------------|--------------------------|

| - Authentification | - Gestion de | - Test | - Authentification | - Authentification |

| - Gestion des Comptes | Comptes | | - Gestion des Comptes | - Gestion des Comptes |

| - Gestion des Animaux | Animaux | | - Gestion des Animaux | - Gestion des Animaux |

| - Gestion des Rôles | Rôles | | - Gestion des Rôles | - Gestion des Rôles |

| - Gestion des Tâches | | | - Gestion des Tâches | - Gestion des Tâches |

| - Rapports Statistiques | | | - Rapports Statistiques | - Rapports Statistiques |

Dans ce tableau :

* **Fonctionnalités Prévues** : Cette colonne recense toutes les fonctionnalités prévues pour l'application, classées par ordre de priorité.
* **En Développement** : Les fonctionnalités en cours de développement sont listées ici. Chaque fonctionnalité est assignée à un développeur.
* **En Test** : Une fois qu'une fonctionnalité est développée, elle est déplacée ici pour être testée.
* **Terminées (Br. Développ.)** : Une fois qu'une fonctionnalité a passé les tests et est prête à être intégrée dans la branche de développement, elle est déplacée dans cette colonne.
* **Mergées (Br. Principale)** : Une fois qu'une fonctionnalité a été intégrée dans la branche principale du dépôt de code, elle est déplacée ici pour indiquer qu'elle est disponible dans la version actuelle de l'application.

Ce tableau nous permet de suivre de manière détaillée l'avancement de chaque fonctionnalité, de détecter les éventuels problèmes et de garantir une progression efficace du projet.

Haut du formulaire