NOM	NAFFER
Prénom	Nicolas
Date de naissance	27/01/1993

Copie à rendre

TP – Développeur Web et Web Mobile

Documents à compléter et à rendre

Lien du git : https://github.com/nafniko/ZooArcadia.git
Lien de l'outil de gestion de projet : https://trello.com/b/8KjN4UHS
Lien du déploiement : Acceuil : Zoo Arcadia
Login et mot de passe administrateur : admin@zoo.com | Azerty13@

Partie 1 : Analyse des besoins

Le projet : Zoo Arcadia

Le projet Zoo Arcadia est une application web et web mobile que j'ai développé à l'aide du cahier des charges du client. l'application permet aux utilisateurs d'avoir une expérience interactive et découvrir ce que tout le zoo peut offrir. Grâce à son design responsive, elle est accessible depuis divers appareils : smartphones, tablettes, ordinateurs portables et de bureau.

Les utilisateurs ont la possibilité de découvrir les services du zoo tels que la visite guidée , la visite en petit train et les restaurants. Ils peuvent consulter la liste des animaux et leurs habitats naturels. Ils peuvent aussi laisser un avis et contacter le zoo.

Le client qui est l'administrateur du site possède une interface pour une gestion globale du site web. Les employés ainsi que les vétérinaires ont un accès à cette interface mais ils ont un accès limité , les permissions dépendent de leur rôle.

L'application est basée sur le langage php, la structure du front end est développé en html5, le style en css (via bootstrap et scss) et l'animation en js via bootstrap.

Pour le back end j'ai utilisé le PDO (Php Data Object) qui permet de gérer la base de données relationnelles. Les données sont stockées dans mariadb.

Pour le NoSQL j'ai utilisé mongodb pour la gestion des avis utilisateurs.

Le déploiement est effectué par Heroku. Heroku garantit la confidentialité des données sensibles du client grâce au variable d'environnement qui stocke les accès au base de données du client.

Le cahier des charges

La page d'accueil

Le client souhaite une page d'accueil attractive pour capter l'attention des visiteurs et leur offrir une vue globale des services, habitats, et animaux proposés par le zoo.

Les spécifications fonctionnelles :

Un article de présentation du zoo avec un texte descriptif et une image.

Un article listant les habitats disponibles (nom,description, images).

Un article présentant les services (nom, description).

Une section des derniers avis validés par un employé et un formulaire pour poster un avis.

La navigation

Description :

Le menu facilite la navigation sur le site en offrant un accès rapide aux sections principales.

Fonctionnalités :

Lien vers la page d'accueil.

Accès à la liste des services.

Accès à la liste des habitats.

Lien de connexion (réservé aux vétérinaires, employés, et administrateurs).

Lien vers la page de contact.

Vue globale de tous les services

Description :

Une page récapitulative présentant tous les services du zoo. Chaque service doit afficher un nom et une description.

Les services disponibles sont :

Restauration, Visite des habitats avec un guide (gratuit), Visite du zoo en petit train

Fonctionnalité :

Les services doivent être modifiables, ajoutés ou supprimés depuis l'espace Administrateur.

Vue globale des habitats

Description :

Une page présentant tous les habitats du zoo sous forme de vignettes (nom et image uniquement).

Fonctionnalités :

Au clic sur un habitat :

Afficher les détails (nom, description, image, liste des animaux).

Chaque animal doit inclure :

Prénom, race, images, et habitat associé.

Au clic sur un animal :

Afficher ses propriétés détaillées.

Inclure les derniers avis du vétérinaire.

Gestion des avis

Description :

Permettre aux visiteurs de laisser un commentaire avec un pseudo et un champ de texte pour l'avis.

Fonctionnalités :

Les avis sont soumis via un formulaire sur la page d'accueil.

Les employés valident ou refusent les avis depuis leur espace.

Une fois validé, l'avis est affiché sur la page d'accueil.

Espace Administrateur

Utilisateur concerné : Administrateur

Description :

L'administrateur gère les comptes, les contenus du site, et accède aux statistiques.

Fonctionnalités :

Gestion des comptes :

Création des comptes "employé" et "vétérinaire" avec adresse email et mot de passe.

Le compte administrateur ne peut pas être créé via l'application.

Gestion des contenus :

Création, modification et suppression des services, horaires, habitats et animaux.

Gestion des comptes rendus :

Visualisation de tous les comptes rendus des vétérinaires.

Possibilité de filtrer les comptes rendus par race ou date.

Dashboard :

Affichage des statistiques sur le nombre de consultations par animal

Espace Employé

Description :

L'espace employé permet de gérer les avis, les services du zoo, et de consigner les consommations alimentaires des animaux**.**

Fonctionnalités :

Gestion des avis :

Valider ou invalider un avis soumis par un visiteur.

Modification des services :

Ajouter, modifier ou supprimer les services proposés par le zoo**.**

Suivi des repas des animaux :

Sélectionner un animal**,** Enregistrer une consommation alimentaire en précisant date, heure, nourriture, quantité.

Espace Vétérinaire

Description :

L'espace vétérinaire permet de suivre la santé des animaux, de saisir des comptes rendus et d'ajouter des commentaires sur les habitats.

Fonctionnalités :

Comptes rendus des animaux :

Remplir un compte rendu quotidien pour chaque animal, en se basant sur l'état de l'animal.

Commentaires sur les habitats :

Ajouter des commentaires sur l'état des habitats et suggérer des améliorations éventuelles.

Historique des consommations alimentaires :

Voir l'historique de ce que chaque animal a mangé, basé sur les saisies effectuées par les employés dans leur espace.

Connexion

Description :

Seuls les utilisateurs ayant un rôle d'administrateur, vétérinaire ou employé peuvent se connecter à l'application.

Fonctionnalités :

Saisie du nom d'utilisateur (email) et du mot de passe :

L'utilisateur doit entrer son email et son mot de passe pour se connecter.

Accès restreint aux utilisateurs autorisés :

Seul un administrateur, un vétérinaire ou un employé peut accéder à l'application via la connexion.

Contact

Description :

Un visiteur peut contacter le zoo via un formulaire accessible depuis le menu de l'application. Le formulaire permet d'envoyer une demande au zoo, et l'employé peut répondre directement par mail.

Fonctionnalités :

Accès au formulaire de contact depuis le menu :

Le visiteur peut accéder à la page de contact depuis le menu de l'application.

Formulaire de contact :

Le formulaire demande les informations Titre de la demande, description, email de l'utilisateur .

Envoi par email :

Une fois le formulaire soumis, la demande est envoyée par email au zoo.

Statistique sur la consultation des habitats

Description :

Chaque fois qu'un visiteur consulte la page d'un animal, le compteur de consultations de cet animal est incrémenté de 1. Ces informations sont stockées dans une base de données non relationnelle et sont utilisées pour générer des statistiques sur les animaux les plus consultés, consultables par l'administrateur via un dashboard.

Fonctionnalités :

Augmentation du compteur de consultations :

Lorsqu'un visiteur consulte la page d'un animal, le nombre de consultations pour cet animal est augmenté de 1.

Stockage dans une base de données non relationnelle :

Les statistiques de consultation sont stockées dans une base de données non relationnelle (par exemple MongoDB), adaptée à ce type de données.

Exploitation des données dans le Dashboard :

Ces données sont récupérées pour être affichées dans un dashboard accessible à l'administrateur. Ce dashboard permet à José de visualiser quels animaux sont les plus populaires en fonction des consultations.

Déploiement de l'application :

L'application doit être déployée pour permettre l'utilisation de ces statistiques en temps réel.

Stack technique

Frontend

HTML5 :

J'ai utilisé HTML5, un langage de balisage essentiel pour structurer le contenu des pages web. Il constitue la base de tout site internet. Je m'en suis servi pour afficher mes pages sur les navigateurs et rendre le contenu accessible à tous.

Bootstrap :

J'ai opté pour Bootstrap, un framework CSS qui simplifie la création de sites web responsives. Grâce à ses composants préconçus (boutons, barres de navigation, formulaires, etc.), j'ai pu gagner du temps et garantir une compatibilité automatique sur différents types d'écrans. De plus, j'ai utilisé ses composants pour leur praticité et leur design soigné.

CSS :

J'ai utilisé CSS pour compléter Bootstrap en personnalisant davantage l'apparence visuelle du site. Cela m'a permis de définir les couleurs, les polices, les espacements et même d'ajouter des animations pour une mise en page plus unique.

JavaScript :

J'ai intégré JavaScript pour rendre mon site interactif et dynamique. Je l'ai utilisé principalement pour rendre certains éléments HTML interactifs au clic et pour transmettre des données à la base de données NoSQL.

Backend

PHP 8.02 :

J'ai utilisé PHP pour développer l'ensemble du backend de l'application. En tant que langage de programmation côté serveur, PHP m'a permis de manipuler facilement les données stockées dans la base relationnelle et de créer des pages dynamiques adaptées aux besoins des utilisateurs.

MariaDB :

J'ai utilisé MariaDB comme système de gestion de base de données relationnelle. Il m'a permis de gérer efficacement les informations du zoo, comme les données sur les animaux, les utilisateurs, les services et les rapports. MariaDB s'intègre parfaitement avec PHP, facilitant ainsi l'exécution de requêtes SQL et la manipulation des données.

Outils supplémentaires

Git :

J'ai utilisé Git pour gérer les versions de mon code. Cet outil me permet de conserver un historique complet des modifications et de revenir facilement à des versions antérieures si nécessaire.

GitHub :

J'ai hébergé mon code sur GitHub pour le sécuriser contre toute perte de données. Cela m'a aussi permis d'accéder à mon travail depuis n'importe quelle machine.

PHPMailer :

J'ai utilisé PHPMailer, une bibliothèque PHP, pour envoyer des e-mails de manière fiable et sécurisée. Cela a été particulièrement utile pour gérer les notifications et les contacts.

OWASP ZAP :

J'ai utilisé OWASP ZAP pour tester les failles de sécurité de mon application, afin d'assurer une meilleure protection contre les attaques potentielles.

Symfony Anti-CSRF Token :

Pour protéger mon application contre les attaques CSRF, j'ai implémenté les tokens Anti-CSRF de Symfony. Cela garantit que seuls les formulaires provenant de sources légitimes peuvent être soumis.

Déploiement

Heroku :

J'ai déployé mon application sur Heroku, une plateforme cloud qui simplifie la mise en production. Son processus de déploiement rapide m'a permis de lier mon dépôt GitHub et de lancer facilement la mise en ligne. J'ai également utilisé l'add-on

JawsDB pour créer une base de données distante et stocker mes données sensibles dans les variables de configuration de la plateforme. J'ai effectué le même processus pour Mongodb avec mongo atlas.

Mise en place de l’environnement de travail pour le projet Zoo Arcadia

Environnement de développement

IDE (Integrated Development Environment) :

J'ai utilisé Visual Studio Code (VS Code) comme IDE pour le développement du projet. VS Code, un éditeur de code source open-source développé par Microsoft, offre une interface fluide et de nombreuses extensions permettant d’ajouter des fonctionnalités spécifiques comme le formatage du code, l'intégration Git, le linting (analyse du code), ou encore le Live Server. Son terminal intégré me permet d'exécuter des commandes sans quitter l’éditeur, ce qui simplifie grandement mon flux de travail.

Serveur local avec XAMPP :

J'ai choisi XAMPP comme environnement de serveur local. Cette distribution open-source contient Apache (serveur web), MySQL/MariaDB (base de données) et PHP. Cela m’a permis de développer, tester et exécuter mon application directement sur ma machine, sans nécessiter un serveur distant. XAMPP offre une solution tout-en-un pratique pour le développement local.

Terminal (ou ligne de commande) :

J'ai régulièrement utilisé le terminal pour interagir avec mon environnement de développement. Il m'a permis de gérer MariaDB, d’installer ou configurer différents composants et frameworks, et d’exécuter des commandes liées à mon projet. Cet outil offre un contrôle puissant et une exécution rapide des tâches complexes.

phpMyAdmin :

J'ai utilisé phpMyAdmin pour gérer visuellement les bases de données MySQL et MariaDB. Grâce à son interface intuitif, j'ai pu tester mes requêtes SQL plus facilement, ce qui m’a aidé à améliorer ma maîtrise de la gestion des bases de données.

Google Chrome et DevTools :

J'ai utilisé Google Chrome comme navigateur principal, car il dispose d’outils de développement intégrés très utiles (Chrome DevTools). Ces outils m'ont permis d’inspecter et de déboguer le code HTML, CSS et JavaScript pour optimiser l’affichage et le comportement de mon site.

Mécanismes de sécurité

Pour assurer la sécurité de mon application, j'ai utilisé plusieurs techniques, outils et bonnes pratiques.

OWASP ZAP :

J'ai installé OWASP ZAP, un outil open-source de détection de vulnérabilités, pour analyser le site Zoo Arcadia et identifier ses failles potentielles. Cet outil m'a fourni des rapports détaillés avec des solutions et des liens vers des documentations. Grâce à ces analyses, j'ai renforcé la sécurité en appliquant les mécanismes suivants :

Protection contre les attaques XSS (Cross-Site Scripting) :

Utilisation de htmlentities() et htmlspecialchars() pour traiter les entrées utilisateur, empêchant l'injection de scripts malveillants dans les pages HTML.

Mise en place d'une Content Security Policy (CSP) pour limiter les ressources externes pouvant être chargées sur le site.

Prévention des injections SQL :

Utilisation de requêtes préparées avec PDO pour interagir avec la base de données. Les données utilisateur sont liées à des paramètres sécurisés, empêchant ainsi les injections SQL.

Sécurisation des sessions :

Utilisation de cookies sécurisés pour les sessions, avec les attributs HttpOnly et Secure activés. Cela limite les risques de vol de session via des attaques XSS.

Hashing des mots de passe :

Les mots de passe des utilisateurs sont hachés avec des algorithmes cryptographiques sécurisés comme bcrypt, garantissant leur protection en cas de fuite de la base de données.

Protection contre les attaques CSRF (Cross-Site Request Forgery) :

Implémentation des tokens anti-CSRF de Symfony dans les formulaires. Ces tokens uniques garantissent que seules les requêtes provenant du formulaire généré par l'application sont validées.

Utilisation de HTTPS :

Le site est déployé avec HTTPS, ce qui garantit que toutes les communications entre le navigateur du client et le serveur sont chiffrées. Ce certificat SSL est fourni par l'hébergeur Heroku, renforçant la confidentialité et l'intégrité des données échangées.

Conclusion

Ces choix technologiques et ces mécanismes de sécurité m'ont permis de créer un environnement de travail stable, efficace et sécurisé pour le développement et le déploiement du projet Zoo Arcadia.

Veille technologique sur les vulnérabilités de sécurité

Lors de la phase d'analyse des failles de sécurité de mon projet, j'ai utilisé OWASP ZAP, un outil spécialisé dans l'identification des vulnérabilités des applications web. Cet outil m'a fourni des rapports détaillés sur les failles détectées, accompagnés de liens vers des ressources fiables pour approfondir chaque problématique. Ces informations m'ont permis d'effectuer une veille technologique ciblée sur les menaces spécifiques à mon projet.

En consultant les ressources fournies, j'ai découvert plusieurs pratiques et outils pour renforcer la sécurité de mon application. Par exemple, j'ai appris l'existence et l'importance des jetons anti-CSRF (Cross-Site Request Forgery) pour protéger les formulaires web contre des attaques malveillantes. À cette occasion, j'ai également découvert que Symfony propose un composant dédié pour gérer ce type de menace. Grâce à ces recherches, j'ai pu intégrer efficacement des protections anti-CSRF dans mon projet, en utilisant les outils proposés par le framework Symfony.

Cette veille m'a non seulement permis de sécuriser mon projet, mais elle a également enrichi mes compétences en matière de cybersécurité, en me sensibilisant davantage aux pratiques essentielles pour développer des applications fiables et robustes.

Recherche

Dans le monde du développement, je me rends compte chaque jour que le développement web est un domaine en constante évolution. Il m'arrive souvent de tomber sur des situations nouvelles qui nécessitent de me remettre en question ou d'apprendre des notions que je ne maîtrise pas encore. Faire des recherches est devenu pour moi un réflexe essentiel pour approfondir mes connaissances, mieux comprendre certains concepts ou trouver des solutions à des problèmes spécifiques. Ce processus d'apprentissage continu m'aide non seulement à surmonter les défis rencontrés, mais aussi à m'améliorer et à évoluer dans mes projets.

Durant le développement de mon projet web Zoo Arcadia, j'ai rencontré une situation où l'utilisation d'un champ <input type="hidden"> était nécessaire pour stocker des données sensibles ou temporaires sans les afficher à l'utilisateur. Cependant, je n'étais pas familier avec l'utilisation de cet élément HTML et de ses attributs associés, ce qui m'a poussé à effectuer des recherches approfondies.

Pour résoudre ce problème, je me suis tourné vers la documentation de Mozilla Developer Network (MDN), une ressource fiable pour les développeurs web. J'y ai appris que l'élément <input type="hidden"> permet de stocker des informations dans un formulaire sans les afficher visuellement sur la page. Cela peut être utile pour gérer des informations comme des identifiants de session ou des tokens d'authentification. En plus de cela, j'ai approfondi mes connaissances sur d'autres attributs des balises <input>, comme name, value, et id, qui sont cruciaux pour manipuler correctement les données envoyées via un formulaire.

Grâce à ma recherche sur MDN, j'ai compris que bien que les données stockées dans un champ <input type="hidden"> ne soient pas visibles, elles peuvent être modifiées par un utilisateur malveillant si des précautions de sécurité ne sont pas prises, comme la validation côté serveur.

Source consultée :

MDN Web Docs - HTML <input> element

Extrait du site (MDN) :

> "The <input type="hidden"> element is used to store data that cannot be seen or modified by the user unless they know the proper URL. It can be useful when sending data that the user doesn't need to be aware of, like an authentication token or a session ID."

Traduction en français :

> "L'élément <input type="hidden"> est utilisé pour stocker des données qui ne peuvent pas être vues ou modifiées par l'utilisateur, sauf si celui-ci connaît l'URL appropriée. Il peut être utile pour envoyer des données dont l'utilisateur n'a pas besoin d'être informé, telles qu'un jeton d'authentification ou un identifiant de session."

Cet extrait m'a permis de comprendre l'usage et les limitations de l'élément <input type="hidden">, et d'adopter des pratiques sécuritaires dans la gestion des données sensibles au sein de mon projet.