6

NOM	Thenault
Prénom	Fanny
Date de naissance	08/10/1987

Copie à rendre

TP – Développeur Web et Web Mobile

Documents à compléter et à rendre

Lien du git : https://github.com/fanny-elimar/ARCADIA.git , branche main
Lien de l'outil de gestion de projet :
https://trello.com/invite/b/665d88bade546439eaf86e83/ATTIf8cb4203d77da42ceea014959362286c3FB6BEB4/arcadia
Lien du déploiement : Arcadia (infinite-tor-98460-50d9abfb93f7.herokuapp.com)
Login et mot de passe administrateur : admin@admin.fr / ArcAdmin !198

SANS CES ELEMENTS, VOTRE COPIE SERA REJETEE

Partie 1 : Analyse des besoins

1. **Effectuez un résumé du projet en français d'une longueur d'environ 20 lignes soit 200 à 250 mots**

   Arcadia est un zoo situé en Bretagne, près de la forêt de Brocéliande. Les animaux sont répartis sur trois habitats : la savane, la jungle et les marais. Le directeur est très soucieux du bien-être des animaux. Chaque matin, le vétérinaire effectue des contrôles sur les animaux afin de s'assurer de leur bonne santé et la nourriture donnée est calculée en fonction des recommandations indiquées par le vétérinaire. Le zoo est également très impliqué dans la protection de l'environnement ; il est d'ailleurs indépendant au niveau de la production d'énergie.

   Le directeur du zoo souhaitait disposer d'une application web pour augmenter l'attractivité de son zoo. Le site présente les habitats du zoo ainsi que les animaux présents, les services proposés par le zoo et ses horaires d'ouverture et il doit permettre aux visiteurs de laisser un avis suite à leur visite ou de contacter le zoo via un formulaire.
   Ce site est également un outil de travail pour le directeur, les employés et le vétérinaire puisqu'il permet la saisie des comptes-rendus du vétérinaire suite à ses visites, ses recommandations concernant l'alimentation des animaux et la saisie des aliments donnés aux animaux par les employés.

2. **Exprimez le cahier des charges, l'expression du besoin ou les spécifications fonctionnelles du projet**

   1) Couleurs et thèmes :

   Le directeur souhaite que le site reflète l'implication écologique du zoo. Le thème et les couleurs devront donc être sobres et en accord avec la nature.

   2) Fonctionnalités :
      a) Pour les visiteurs :

Sur la page d'accueil, le visiteur doit pouvoir visualiser quelques images du zoo et une présentation de celui-ci, les services proposés (restauration, petit train, visite guidée), les habitats et les animaux que possède le zoo, les avis laissés par les visiteurs précédents.

Le menu doit permettre un accès direct aux éléments suivants : accueil, habitats, services, contact.

Sur la page des services, le visiteur doit pouvoir visualiser l'ensemble des services proposés et les informations correspondantes.

Concernant les habitats, une première page doit permettre une visualisation de tous les habitats. Au clic sur un habitat, le visiteur pourra visualiser une description plus complète de l'habitat et l'ensemble des animaux présents dans cet habitat. Le visiteur aura accès au nom de l'animal, à son espèce, à une ou des images, à son habitat et à son état suite à la visite du vétérinaire.

Sur la page d'accueil, le visiteur peut laisser un avis suite à sa visite. Pour cela, il devra simplement indiqué un pseudonyme. L'avis sera affiché uniquement s'il est validé par un employé.

Un visiteur peut contacter le zoo en utilisant un formulaire de contact qui enverra un mail au gestionnaire. Pour cela, il devra fournir une adresse mail, un titre et un message.

Si le visiteur essaie de se connecter, il sera renvoyé automatiquement à la page d'accueil.

b) Pour les employés :

Les employés doivent pouvoir se connecter à leur espace en fournissant leur adresse mail et leur mot de passe.

Une fois connecté, l'employé doit pouvoir valider ou invalider les avis des visiteurs, modifier les informations concernant les services du zoo consulter les recommandations du vétérinaire et saisir la nourriture donnée aux animaux (type de nourriture, quantité, date et heure).

c) Pour le vétérinaire :

Le vétérinaire doit pouvoir se connecter à son espace en fournissant son adresse mail et son mot de passe.

Une fois connecté, le vétérinaire saisit les informations suite à sa visite de chaque animal : état de l'animal, recommandation de nourriture (type et quantité), date de passage, détail de l'état de l'animal (si nécessaire). Le vétérinaire a aussi la possibilité de laisser un commentaire sur l'état de l'habitat.

Le vétérinaire peut également consulter la nourriture qui a été donnée aux animaux (information saisie par les employés).

d) Pour l'administrateur :

L'administrateur a accès à l'espace gestion du site. Il doit pouvoir effectuer les actions suivantes :

- Création de compte employé ou vétérinaire : il fournira l'email de l'utilisateur et un mot de passe. Le nouvel utilisateur recevra alors un mail indiquant son nom d'utilisateur, qui sera son adresse mail. Il devra se rapprocher de l'administrateur pour connaitre son mot de passe.
  Il ne sera pas possible de créer un autre compte administrateur.
- Modification des services : création, mise à jour, suppression.
- Modification des horaires : création, mise à jour, suppression.
- Modification des habitats : création, mise à jour, suppression.
- Modification des animaux : création, mise à jour, suppression.
- Visualisation des comptes-rendus du vétérinaire avec des filtres permettant de sélectionner un animal ou une date.
- Suivi du nombre de consultation par animal : au clic sur un animal, son compteur est incrémenté de 1. Le directeur du zoo doit pouvoir visualiser le nombre de clic sur chaque animal.

Partie 2 : Spécifications technique

1. **Spécifiez les technologies que vous avez utilisées en justifiant les conditions d'utilisation et pourquoi le choix de ses éléments**

Les technologies utilisées pour ce projet sont les suivantes :

- Front :
  - HTML 5,
  - CSS et Bootstrap : bootstrap permet une mise en page simple, rapide et responsive.
  - Javascript : javascript a été utilisé pour créer des affichages dynamiques, notamment via la librairie Popper pour les carrousels et les affichages des avis. Javascript a également été utilisé pour les redirections automatiques, pour filtrer les résultats de recherche lors de l'affichage des comptes-rendus du vétérinaire par l'administrateur et pour l'affichage du graphique des clics par animaux.
- Back :
  - PHP version 8.2 avec PDO pour la connexion à la base de données.
  - Base de données relationnelle : PostGreSQL version 16.0
  - Base de données non relationnelle : MongoDB Atlas
- Déploiement :
  - Le site a été déployé via Heroku et la base de données PostGreSql a été mise en ligne sur AlwaysData.

2. **Comment avez-vous mis en place votre environnement de travail ? Justifiez vos choix. (README.md)**

Afin de réaliser ce projet, j'ai utilisé VSCode.
J'ai utilisé un serveur local avec Apache sous XAMPP.
La base de données a été créée sous PostGreSql et les requêtes SQL ont été réalisées avec PSQL ou pgAdmin4.
La base de données non relationnelle a été créée sur MongoDb Atlas.

Le fichier README.md indique les fichiers à télécharger et installer afin d'avoir accès au projet.

3. **Énumérez les mécanismes de sécurité que vous avez mis en place, aussi bien sur vos formulaires que sur les composants front-end ainsi que back-end.**

<u>Faille XSS :</u>
Les failles XSS sont une injection de code malveillant dans un formulaire, permettant par exemple d'exécuter un code Javascript dans le navigateur de la victime. Pour cela, l'utilisateur malveillant entre un code malveillant dans un

champ du formulaire. Ce code est ensuite enregistré dans la base de données. Si cette donnée est affichée sur le site, le navigateur va lire le code injecté par l'utilisateur malveillant.

➔ Fonction htmlspecialchars() :

Pour se prémunir de ces attaques, il faut donc convertir les caractères spéciaux propres au code HTML ou Javascript tels que les chevrons ou les guillemets en code HTML, en utilisant la fonction htlmspecialchars() lors de l'affichage de données fournies par un utilisateur. Htmlspecialchars() est donc utilisé lorsque l'on affiche des données de la base de données qui ont été entrées par l'utilisateur ou lorsque l'on récupère les données de l'url.

Injections SQL :

Lors d'injection SQL, l'attaquant injecte du code malveillant dans les requêtes SQL afin de récupérer ou d'altérer les données de la base de données. Pour s'en prémunir, il est recommandé d'utiliser des requêtes préparées.

➔ Requêtes préparées :

Avec une requête préparée, les données et la requête sont envoyées séparément au serveur. Le serveur exécute dans un premier temps la requête SQL avec les marqueurs, puis les marqueurs sont liés aux données. Ainsi, si du code malveillant a été injecté dans les données (via un formulaire par exemple), il ne sera pas exécuté.

Pages protégées :

Pour les pages uniquement accessibles aux utilisateurs connectés (administrateur, vétérinaire ou employé), une vérification de la session est mise en place en haut de chaque page, avec une redirection automatique vers la page de connexion ou l'accueil du site.

Mots de passe :

En suivant les recommandations de l'ANSSI, les mots de passe doivent contenir au moins 12 caractères, de types différents (lettres minuscules, majuscules, chiffres et caractères spécieux).

Des vérifications sont donc mises en place :

- Côté client : En HTML, les attributs « minlength » et « required » sont utilisés ;

  Ces vérifications peuvent être contournées par des personnes malveillantes, c'est pourquoi des vérifications côté serveur, en PHP, sont également mises en place :

- Côté serveur : Afin de s'assurer que le mot de passe est sécurisé, la fonction preg_match() est utilisée avec une expression Regex. Ainsi, on vérifie que le mot de passe contient bien 12 caractères dont au moins une majuscule, une minuscule, un chiffre et un caractère spécial. De plus, pour éviter une erreur de l'admin lors de la création du mot de passe, il doit le retaper dans un deuxième champ. Une vérification est réalisée côté serveur pour vérifier que les deux mots de passe sont bien identiques.

Si pour l'aspect sécurité, les vérifications côté serveur seraient suffisantes, une vérification côté client rend l'expérience utilisateur plus agréable car, en cas de non respect des règles, un message d'erreur s'affiche avant l'envoi du formulaire.

4. **Décrivez une veille technologique que vous avez effectuée, sur les vulnérabilités de sécurité.**

   Je me suis créé une page sur Feedly me permettant de suivre les actualités de plusieurs sites de relatif au développement.
   Concernant la sécurité, je suis le bulletin d'actualité du Centre Gouvernemental de veille, d'alerte et de réponse aux attaques informatiques (CERT-FR).
   Les différents sites que je suis mentionnent également les failles de sécurité repérées ou les nouveautés concernant la sécurité.

Partie 3 : Recherche

1. **Décrivez une situation de travail ayant nécessité une recherche durant le projet à partir de site anglophone. N'oubliez pas de citer la source**
   J'ai effectué une recherche sur stackoverflow.com afin de savoir comment rendre le lien du menu correspondant à la page actuelle actif. J'ai utilisé les éléments fournis sur cette page : [css - How to set current page "active" in php - Stack Overflow]

2. **Mentionnez l'extrait du site anglophone qui vous a aidé dans la question précédente en effectuant une traduction en français.**
   Voici des extraits des réponses qui m'ont aidées :

> « first you can read the current filename of the php file you request by
> using $_SERVER['PHP_SELF'] or $_SERVER['REQUEST_URI'] or any other $_SERVER
> global variables that you can use to read your current page and compare it with the link's url,
> something like this
> <a href="offnungszeiten.php" <?php if($_SERVER['PHP_SELF']=='offnungszeiten.php'){
> ?>class="activatepage" <?php } ?> >
> Öffnungszeiten
> »

Traduction : Premièrement, vous pouvez récupérer le nom du fichier php appelé en utilisant $_SERVER[`PHP_SELF'] ou $_SERVER[`REQUEST_URI'] ou n'importe quelle autre variable globale de $_SERVER qui permette de lire le nom de la page actuelle, puis vous la comparez avec le lien de l'url, quelque chose comme ça : [code ci-dessus].

Sur cette page, j'ai également trouvé comment récupérer uniquement la fin de mon URL :

> « this method gets the current page using php […].
> function active($currect_page){
> $url_array = explode('/', $_SERVER['REQUEST_URI']) ;

```
  $url = end($url_array);
  if($currect_page == $url){
      echo 'active'; //class name in css
  }
}
[…]»
```

Traduction : Cette méthode récupère la page actuelle avec php […].
Je n'ai pas utilisé cette fonction directement mais uniquement les méthodes explode() et end().

J'ai ensuite modifié le code pour qu'il corresponde à ce que je souhaitais :
En haut de ma page header.php :

```php
$url_array = explode('/',$_SERVER['PHP_SELF']);
$url = end($url_array);
```

Pour chaque lien de mon menu :

```php
<a href="habitats.php" class="nav-link px-2 <?php
if(str_contains($url,'habitat') || str_contains($url,'animal')){?>active
<?php } ?>">Les habitats</a>
```

J'ai utilisé la méthode str_contains() car je voulais que le lien du menu (ici Habitats) reste actif pour toutes les pages de la « catégorie » habitat (page décrivant tous les habitats, pages de chaque habitat détaillé, pages des animaux).

Dans le fichier css, j'ai ajouté l'élément suivant :

```
.active {
        text-decoration : underline ;
}
```

Partie 4 : Informations complémentaire

1. **Autres ressources**

2. **Informations complémentaires**

Les documents demandés (Manuel d'utilisation, documentation technique, base de donnée, …) sont présents dans le dépôt git, dans le dossier PROJET.