DOSSIER PROJET

Benoît Du Breul de Saconay

Titre professionnel Développeur Web et Web Mobile

Application Web ZOO Arcadia

SOMMAIRE

1. **Résumé du projet** **page 5**

- Présentation du projet "Le Zoo Arcadia" et de ses objectifs.
- Public cible : familles, enseignants, amateurs d'animaux, communauté locale et touristes.

2. **Cahier des Charges** **page 6**

- Définition des besoins et des attentes pour le projet.
- Description des fonctionnalités et des services à offrir.

3. **Diagrammes du Projet Acadia** **page 11**

- Diagramme de classe
- Diagramme de séquence
- Diagramme de cas d’utilisation

4. **Gestion et Développement du Projet Web** **page 16**

Technologies Utilisées

- HTML : Structure du contenu.
- CSS : Stylisation des éléments et mise en forme moderne et responsive.
- JavaScript : Ajout de fonctionnalités interactives.
- PHP : Gestion des connexions et des sessions utilisateurs.
- MySQL : Gestion des données relatives aux animaux, avis, habitats, soins, repas et utilisateurs.
- NoSQL : Base de données pour les clics réseaux sociaux.

5. **Styles et Design** **page 35**

- Charte graphique pour Le Zoo Arcadia.

6. **Maintenance et Support** **page 38**

- Stratégies de sécurité du site.

- Application des mesures de sécurité dans le code

7. **Documentation Technique du Projet "Le Zoo Arcadia"** **page 43**

- Réflexion initiale sur le projet.
- Fonctionnalités clés
- Détails des technologies utilisées
- Processus de déploiement en local et sur O2Switch. Configuration du site en ligne.

8. **Conclusion** **page 50**

- Résumé des objectifs atteints et perspectives d'avenir.

9. **Annexes** **page 51**

- Maquettes
- Manuel d’utilisation
- Jeu d'essai et veille technologique

Compétences du référentiel couvertes par le projet

Pour l'activité type 1 « Développer la partie front-end d'une application web ou web mobile sécurisée »

- Maquetter des interfaces utilisateur web ou web mobile
- Réaliser des interfaces utilisateur statiques web ou web mobile
- Développer la partie dynamique des interfaces utilisateur web ou web mobile

Pour l'activité type 2 « Développer la partie back-end d'une application web ou web mobile sécurisée »

- Mettre en place une base de données relationnelle
- Développer des composants d'accès aux données SQL et NoSQL
- Développer des composants métier côté serveur

Résumé du projet

Arcadia est un zoo prestigieux situé en Bretagne, à proximité de la forêt de Brocéliande, et ouvert au public depuis 1960. Ce parc zoologique se distingue par sa diversité d’animaux répartis selon différents habitats, tels que la savane, la forêt, et plus encore. Chaque jour, une équipe de vétérinaires s’assure du bien-être des animaux en réalisant des contrôles rigoureux et en ajustant précisément la nourriture pour chaque espèce, conformément aux recommandations médicales. Grâce à une gestion efficace et à une attention constante portée à la santé animale, Arcadia prospère financièrement, et les animaux y sont heureux, ce qui fait la fierté de son directeur, José. Ce dernier nourrit de grandes ambitions pour l'avenir du zoo.

Le site web d'Arcadia est conçu pour offrir une navigation fluide aux visiteurs comme aux professionnels. La page d'accueil présente le zoo avec des images évocatrices et des informations sur les habitats et les services disponibles. Un menu de navigation permet d'accéder facilement à toutes les sections du site, y compris les habitats, les services, et les options de contact. Les visiteurs peuvent explorer les différents habitats et découvrir les animaux qui y résident. Une vue d'ensemble des services, comme la restauration et les visites en petit train, est également disponible.

Pour les vétérinaires et les employés, des espaces dédiés permettent la gestion des soins aux animaux, la saisie des données de suivi, et l'enregistrement des consommations alimentaires. Les administrateurs ont accès à des outils de gestion avancés, y compris des statistiques sur la fréquentation des habitats et des comptes rendus des vétérinaires. Les visiteurs peuvent également laisser des avis et contacter le zoo via divers canaux.

Le site web d'Arcadia assure une interaction dynamique et transparente, facilitant la gestion quotidienne tout en offrant aux visiteurs une expérience enrichissante et informative.

Cahier des charges

Arcadia est un zoo situé en France près de la forêt de Brocéliande, en bretagne depuis 1960. Ils possèdent tout un panel d'animaux, réparti par habitat (savane, foret etc) et font extrêmement attention à leurs santés. Chaque jour, plusieurs vétérinaires viennent afin d'effectuer les contrôles sur chaque animal avant l'ouverture du zoo afin de s'assurer que tout se passe bien, de même, toute la nourriture donnée est calculée afin d'avoir le bon grammage (le bon grammage est précisé dans le rapport du vétérinaire).

Le zoo, se porte très bien financièrement, les animaux sont heureux. Cela fait la fierté de son directeur, José, qui a de grandes ambitions.

Composition du site web:

US 1 : Page d'accueil

Utilisateur concerné : Visiteur

La page d'accueil doit comporter :

- Présentation du zoo en y incorporant quelques images
- Mentionnez les différents habitats, services ainsi que les animaux que possède le zoo Les avis du Zoo

US 2 : Menu de l'application

Utilisateur concerné : Visiteur

Le menu est essentiel à l'application, il permet de faciliter la navigation et de fluidifier le trafic, ainsi, il doit composer au minimum :

- Retour vers la page d'accueil

- Accès à tous les services
- Accès à tous les habitats
- Connexion (la connexion sera possible uniquement pour les vétérinaires, employé et administrateur)
- Contact

US 3 : vue globale de tous les services

Utilisateur concerné : Visiteur

Une vue globale est nécessaire afin de proposer une interface simple et récapitulative de tous les services que propose le parc.

Pour le moment, le parc dispose de plusieurs services : restauration, visite des habitats avec un plan, visite du zoo en petit train

US 4 : vue globale des habitats

Utilisateur concerné : Visiteur

Cette page, doit mentionner tous les habitats que propose le zoo et les animaux associés. José vous propose une vue tout d'abord de tous les habitats (en affichant que l'image et le nom), puis, au clic , on affiche le détail avec les animaux ainsi que la description.

Un habitat est caractérisé par :

- Un nom
- Une ou des images
- Une description de l'habitat
- Une liste d'animaux

Un animal est caractérisé par :

- Un Prénom
- Une race

- Une ou des images
- Un habitat où il est affecté

Le vétérinaire, passe régulièrement et saisie des informations depuis son espace sur un animal donné en mentionnant :

- L'état de l'animal
- Le grammage de la nourriture
- Date de passage

US 5 : Avis

Utilisateur concerné : Visiteur

Un visiteur, peut laisser un commentaire s'il le désire. Ce commentaire, contiendra juste un pseudo ainsi qu'un champ "avis" texte.

US 6 : Espace Administrateur

Utilisateur concerné : Administrateur

L'admin, peut visualiser les clics sur les habitats.

Ensuite, l'espace administrateur doit avoir un emplacement avec tous les comptes rendus du vétérinaire, des employés et des siens. Ils seront triés par date.

En effet, il doit pouvoir enregistrer les informations sur les animaux: sélection de l'animal, Etats, nourriture, dates de suivi.

US 7 : Espace Employé

Utilisateur concerné : employé

L’employé, va donner l’alimentation quotidienne aux animaux, il va ainsi depuis son espace, pouvoir sélectionner un animal et lui ajouter une consommation de nourriture, pour cela il va devoir donner la date ainsi que la quantité nourriture.

US 8 : Espace Vétérinaire

Utilisateur concerné : vétérinaire

Un vétérinaire passe quotidiennement dans le zoo, ainsi, depuis son espace, il remplira les états par animaux ainsi que nourriture et date de suivi.

Le vétérinaire voit également sur son espace et par animal, tout ce que l’animal a pu manger et à qu’elle date via la saisie de l’employé sur son espace.

US 9 : Connexion

Utilisateur concerné : Administrateur, vétérinaire, employé

Seul une personne de type administrateur, vétérinaire ou encore employé peut se connecter.

Un visiteur ne peut pas devenir utilisateur, et donc, ne peux pas se crée de compte.
Pour se connecter, il devra juste saisir son identifiant suivi de son mot de passe.

US 10 : Contact

Utilisateur concerné : Visiteur

Un visiteur peut contacter le zoo s’il le souhaite, pour cela, il devra accéder à la page contact depuis le menu applicatif. Pour donner suite à cela, il aura accès à une page avec les réseaux sociaux de contact du zoo ainsi que le mail de contact du zoo. L'employé peut répondre à la demande directement par mail.

Au clic sur l’un des deux réseaux sociaux disponibles un enregistrement se efra dans une base de données non relationnelle et visible sur la page admin,

US 11 : Statistique sur la consultation des habitats

Utilisateur concerné : Visiteur

Un visiteur, quand il cliquera sur l'animal, va augmenter la consultation de 1 pour l'animal donné.

Ces données sont exploitées dans le Dashboard administrateur afin que José puisse visualiser quels animaux plaisent le plus.

1. Diagramme de classes

Le **diagramme de classes** du Zoo Arcadia présente la structure des entités principales de l'application, notamment les utilisateurs (administrateurs, vétérinaires, employés) et les animaux. Chaque entité est définie par des attributs et des méthodes :

- **Utilisateur** :
  - Attributs : utilisateur_id, username, password, role
  - Méthodes : login(), logout()
- **Administrateur** :
  - Méthodes : manageHabitats(),NourrirAnimal(), SelectionAnimal(),EnregistrerHealth()
- **Employé** :
  - Méthodes : NourrirAnimal(), EntrerDate(), SelectionAnimal()
- **Vétérinaire** :
  - Méthodes : SelectionAnimal(), EnregistrerHealth(),NourrirAnimal()
- **Animal** :
  - Attributs : animal_id, nom et espece, health_status
  - Méthodes : updateHealthStatus(), addEnregistrementNourriture()

Le diagramme montre comment les utilisateurs interagissent avec les animaux via des méthodes spécifiques. Les relations entre classes sont définies par des associations, représentant les interactions possibles.

2. Diagramme de séquence

Le **diagramme de séquence** pour le Zoo Arcadia illustre le flux des interactions entre un utilisateur (employé, vétérinaire ou administrateur) et le système de gestion du zoo. Il montre les étapes de connexion, la validation des identifiants et la modification des informations sur les animaux.

- **Employé ou vétérinaire ou administrateur** se connecte via une interface de connexion.
- Après validation, l'utilisateur accède aux informations de la base de données.
- Les actions suivantes incluent la mise à jour des informations sur l'état de santé ou l'alimentation des animaux.
- Le système renvoie une confirmation de l'action effectuée (modifications, validation des informations).

Ce diagramme met en évidence la logique d’interaction étape par étape entre l'utilisateur et le serveur.

3. Diagramme de cas d'utilisation

Le **diagramme de cas d'utilisation** décrit les fonctionnalités principales de l'application et les interactions entre les différents types d'utilisateurs et le système. Il inclut plusieurs **cas d'utilisation** :

- **Visiteur** : Peut consulter les habitats au clic, laisser un avis, contacter le zoo ou visiter le site web.

- **Employé** : Sélectionner un animal, gérer l'alimentation des animaux et la date de suivi.

- **Vétérinaire** : Sélectionner un animal ,Saisie l'état de l’animal, l'alimentation de l‘animal et la date de suivi. Il a accès au suivi des animaux sous forme de tableau.

- **Administrateur** :Sélectionner un animal ,Saisie l'état de l’animal, l'alimentation de l‘animal et la date de suivi. Il a accès au suivi des animaux sous forme de tableau. Il peut également faire le suivi des clics de l’espace habitats.

Diagramme de classe

Diagramme de séquence

Benoit Dbds | September 6, 2024

Crée un diagramme de cas d'utilisation :

- Vis...

Gestion et développement du Projet Web Arcadia

Table des Matières

1. Introduction
2. Développement
3. Styles et Design
4. Sécurité
5. Bonnes Pratiques
6. Ressources et Références

Introduction

Ce document a pour but de fournir des directives claires et précises sur la gestion et le développement du projet web. Il comprend des informations sur la structure du projet, les configurations nécessaires, les pratiques de développement, le déploiement et la maintenance.

Structure de Code pour Le Zoo Arcadia

1. HTML : Structure du contenu

Ce code HTML crée une page d'accueil pour le site du zoo Arcadia, présentant le logo et un menu de navigation en en-tête. Le corps principal affiche des images et des descriptions du zoo, mettant en avant ses valeurs écologiques, la diversité animale et les attractions disponibles. Une section dédiée aux avis des visiteurs est incluse, ainsi qu'une présentation des services comme la visite en petit train et la restauration sur place. Enfin, le pied de page propose des images supplémentaires et des liens vers les mentions légales et la politique de données personnelles.

```html
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/main.css">
    <link rel="stylesheet" href="css/accueil.css">
    <script defer src="js/main.js"></script>
    <title>Le Zoo Arcadia</title>
</head>

<body>
    <header>
        <nav class="navbar">
            <div id="cover">
                <a href="index.html">
                    <img src="https://e-solution.timyx.com/wp-reseau/benoit/wp-content/uploads/sites/13/2024/08/logo-arcadia-zoo-benoit.svg"
                        alt="Logo Arcadia Zoo" class="accueil">
                </a>
            </div>
            <nav class="Header">
                <a href="connexions.php">Connexion employés</a>
                <a href="services.html">Services</a>
                <a href="habitats.php">Habitats</a>
                <a href="contact.html">Contact</a>
                <a href="avis.php">Avis</a>
            </nav>
            <button type="button" class="Menu">Menu</button>
        </nav>
    </header>

    <main>
        <img src="https://cdn.pixabay.com/photo/2018/06/30/19/02/panda-3508153_1280.jpg" alt="Panda"
            class="image-panda">
        <div class="text-panda">
```

```html
<div class="text-panda">
    <p>Le zoo d'Arcadia est un zoo à vocation ecoresponsable, le respect des animaux et de leurs habitats
        naturels est une priorité. Entièrement indépendant en énergie, nous sommes fiers de nos valeurs
        écologiques.</p>
    <p>Le zoo met tout en œuvre pour développer et soutenir des actions de conservation pour la préservation de
        la biodiversité mondiale. Son objectif est de sensibiliser le grand public et les visiteurs du ZooParc
        Arcadia en particulier à la nécessité de ces actions.</p>
</div>
<div class="flex-container">
    <div class="explicatif">
        <p>Plus de 150 animaux sauvages vivent paisiblement en semi-liberté sur 80 hectares de forêts, plaines
            et vallons. Tigres, lions, pandas, girafes, partez à la découverte de 120 espèces des 5 continents à
            travers 2 circuits de visite à pied et observez les animaux dans leurs habitats naturels. Ici, tout
            a été conçu pour le bien-être des animaux. Le parc propose également plusieurs attractions pour
            petits et grands. Une aventure exceptionnelle au coeur de Brocéliande à vivre en famille ou entre
            amis.</p>
    </div>
    <div class="image-container">
        <img src="https://cdn.pixabay.com/photo/2019/09/17/20/47/prague-4484518_1280.jpg" alt="Girafes du parc"
            class="image-girafes">
    </div>
</div>

<img src="https://cdn.pixabay.com/photo/2019/12/02/13/54/broceliande-4667802_1280.jpg" alt="Broceliande"
    class="image-broceliande">

<ul class="horaires">
    <li class="title">Nos horaires</li>
    <li><span class="day">Lundi</span> <span class="time">08:00 - 18:00</span></li>
    <li><span class="day">Mardi</span> <span class="time">08:00 - 18:00</span></li>
    <li><span class="day">Mercredi</span> <span class="time">08:00 - 18:00</span></li>
    <li><span class="day">Jeudi</span> <span class="time">08:00 - 18:00</span></li>
    <li><span class="day">Vendredi</span> <span class="time">08:00 - 18:00</span></li>
    <li><span class="day">Samedi</span> <span class="time">08:00 - 18:00</span></li>
```

```html
        <li><span class="day">Dimanche</span> <span class="time">08:00 - 18:00</span></li>
    </ul>

    <div class="avis">
        <h2>Votre avis compte pour nous!</h2>
    </div>

    <div class="articles-container">
        <article>
            <h4>Martine C :</h4>L'accueil au top ! Les activités pédagogiques au top ! L'organisation est bien
            gérée. Le tarif vaut la sortie. Je vous recommande!
        </article>
        <br>
        <article>
            <h4>Vanessa R :</h4>Très beau parc ombragé, animaux magnifiques, de beaux espaces, le nécessaire pour se
            restaurer, très chouette visite en famille !
        </article>
        <br>
        <article>
            <h4>Mickael T :</h4>Une visite des plus merveilleuses comme à chaque fois ! Les vastes enclos, les
            paysages, les animaux qui semblent être heureux ! Un régal ! Sans compter les nouveaux aménagements qui
            sont très bien pensés ! Bravo !
        </article>
    </div>
    <br>
</main>

<div id="services-container">
    <div class="service-item">
        <a href="services.html">
            <img src="https://cdn.pixabay.com/photo/2017/08/29/04/40/tannheim-2692124_1280.jpg"
                alt="petit train du parc" class="image-train">
            <p href="services.html" class="description">Visite en petit train</p>
```

```html
            </a>
        </div>
        <div class="service-item">
            <a href="services.html">
                <img src="https://cdn.pixabay.com/photo/2018/05/22/23/14/burger-3422815_1280.jpg" alt="restauration"
                    class="image-restauration">
                <p class="description">Restauration sur place</p>
            </a>
        </div>
    </div>

    <footer>
        <div id="imgfooter">
            <img src="https://cdn.pixabay.com/photo/2018/11/11/16/51/bird-3809147_1280.jpg" alt="oiseau"
                class="imgbasg">
            <img src="https://cdn.pixabay.com/photo/2021/12/31/11/51/penguin-6905568_1280.jpg" alt="pingouin"
                class="imgbasd">
        </div>
        <div class="Mentions">
            <a href="mentions.html">Mentions légales</a>
            <a href="rgpd.html">Politique de données personnelles</a>
        </div>
    </footer>
</body>

</html>
```

2. CSS : Stylisation des éléments

- **Utilisation des classes et ID** :
    - **Classes** : Appliquez des styles réutilisables à plusieurs éléments (.button, .title).
    - **ID** : Pour les éléments uniques qui nécessitent un style spécifique.

Ce code CSS met en forme la page web avec un arrière-plan fixe et centré, des éléments de navigation stylisés avec flexbox, des effets visuels comme des ombres et des transformations, tout en s'adaptant aux différentes tailles d'écrans grâce aux media queries.

Voicile code CSS pour la mise en page du header de toutes mes pages. Il est integré dans la page globale main.css :

```css
        /*header*/
html, body {
   padding: 0;
   margin: 10px 8px 10px 10px;
   font-family: Geneva, Tahoma, sans-serif;
}

body {
   background-image: url('https://e-solution.timyx.com/wp-reseau/benoit/wp-content/uploads/sites/13/2024/08/fond-feuille.svg');
```

```
    background-size: cover; /* Ajuste l'image pour couvrir tout l'arrière-plan */
    background-position: center; /* Centre l'image */
    background-attachment: fixed; /* Fixe l'image d'arrière-plan */
    background-repeat: no-repeat; /* Empêche la répétition de l'image */
}

#cover {
    width: 200px;
    min-width: 200px;
    height: 150px;
    border-radius:10px;
    display: flex;
    align-items: center;
    justify-content: center;
    margin-right: 15px;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
    filter: drop-shadow(4px 4px 6px rgba(0, 0, 0, 0.4));
}

.navbar {
    display: flex;
    align-items: center;
    justify-content: center;
}

.accueil {
    font-size: x-large;
    font-weight: bold;
    color: black;
}

a:hover {
    font-style: oblique;
}

.Header {
    font-size: x-large;
    display: flex;
    justify-content: center;
    align-items: center;
    gap: 60px;
    width: 100%;
    max-width: 100%;
    height: 20vh;
    border-radius:10px;
    z-index: 1000;
    background-color: #445843;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
    box-shadow: 4px 4px 6px rgba(0, 0, 0, 0.4);
```

```
}

.Header a:active {
    transform: translateY(2px);
}

.Menu {
    display: none;
    border-radius:10px;
    width: 100px;
    margin: 20px;
    background-color: #445843;
    cursor: pointer;
    height: 10vh;
    justify-content: center;
    align-items: center;
    color: white;
    font-size: large;
    z-index: 1002;
    text-shadow: 3px 3px 4px rgba(0, 0, 0, 0.3);
    box-shadow: 4px 4px 6px rgba(0, 0, 0, 0.4);
}

@media (max-width: 1100px) {
    .Menu {
        display: flex;
        justify-content: center;
        align-items: center;
    }

    .Header {
        display: none;
        flex-direction: column;
        position: fixed;
        top: 178px;
        left: 0;
        right: 0;
        bottom: 0;
        background-color: #445843;
        padding: 20px;
        align-items: center;
        justify-content: flex-start;
    }

    .Header.active {
        display: flex;
        gap: 0;
        margin: 0 0 0 20px;
        background-color: #5d7854;
```

```
    height: 160px;
    width: calc(100% - 75px);
  }

  .Header a {
    color: white;
    margin: 0;
    font-size: 1em;
  }
}

@media (max-width: 480px) {
  .Header.active {
    display: flex;
    gap: 0;
    width: calc(100% - 75px);
    top: 183px;
  }
}
```

3. JavaScript : Interactions utilisateur et logique de l'application

- **Implémentation des interactions** :

  - **Événements de clic** : Pour le bouton menu et les boutons habitats, le menus déroulants.

Main.js va venir gérer le clic du bouton du menu qui apparaît dans le cas d'une ouverture de la page web en version mobile et petits écrans :

```
JS main.js > ...
document.addEventListener("DOMContentLoaded", () => {
  const menuButton = document.querySelector(".Menu");
  const headerNav = document.querySelector(".Header");

  menuButton.addEventListener("click", () => {
    headerNav.classList.toggle("active");
  });
});
```

habitats,js gère deux fonctionnalités principales pour une page web :

1. **Menu de navigation** : Lorsqu'un bouton de menu (.Menu) est cliqué, il bascule la classe active sur l'élément de navigation (.Header), ce qui est généralement utilisé pour afficher ou masquer un menu.

2. **Affichage de contenu** : Pour chaque bouton spécifique (défini par des classes comme .foret, .savane, etc.), lorsqu'il est cliqué, le script empêche le comportement par défaut du bouton, envoie les données d'un formulaire via AJAX (si le contenu associé n'est pas déjà visible), puis bascule la visibilité du contenu associé (défini par des classes comme .liste-cachee, .liste-cachee-deux, etc.) en ajoutant ou retirant la classe liste-visible.

```javascript
document.addEventListener("DOMContentLoaded", () => {
  function toggleListe(buttonClass, listeClass) {
    document.querySelector(buttonClass).addEventListener("click", function (e) {
      e.preventDefault(); // Empêche le comportement par défaut du bouton
      const liste = document.querySelector(listeClass);
      const isVisible = liste.classList.contains("liste-visible"); // Vérifie si c'est déjà visibl

      if (!isVisible) {
        // Soumission du formulaire via AJAX si le contenu n'est pas déjà visible
        const form = this.closest("form");
        const formData = new FormData(form);

        fetch(form.action, {
          method: form.method,
          body: formData,
        })
          .then((response) => response.text())
          .then((data) => {
            // Vous pouvez ajouter ici du code pour manipuler le DOM ou afficher un message de suc
            console.log("Formulaire soumis avec succès");
          })
          .catch((error) => {
            console.error("Erreur:", error);
          });
      }

      // Toggle de la visibilité
      liste.classList.toggle("liste-visible");
    });
  }

  // Initialiser les boutons avec leurs listes correspondantes
  toggleListe(".foret", ".liste-cachee");
  toggleListe(".savane", ".liste-cachee-deux");
  toggleListe(".foret-montagneuse", ".liste-cachee-trois");
  toggleListe(".antarctique", ".liste-cachee-quatre");
});
```

- **Animations** : , Cartes interactives service : Cela a permis d'ajouter des transitions et effets pour une expérience utilisateur fluide.

  Ce code JavaScript permet d'ajouter et de retirer dynamiquement une classe "active" aux éléments avec la classe .card lorsque ceux-ci sont touchés sur un appareil tactile, tout en retirant la classe "active" des autres cartes et en désactivant toutes les cartes si une zone extérieure est touchée.

```
services.js > ...
document.addEventListener("DOMContentLoaded", () => {
  const menuButton = document.querySelector(".Menu");
  const headerNav = document.querySelector(".Header");

  menuButton.addEventListener("click", () => {
    headerNav.classList.toggle("active");
  });

  const cards = document.querySelectorAll(".card");

  cards.forEach((card) => {
    card.addEventListener("touchstart", function () {
      cards.forEach((c) => c.classList.remove("active")); // Retirer la classe active des autres cartes
      card.classList.add("active"); // Ajouter la classe active à la carte touchée
    });
  });

  document.addEventListener("touchstart", function (event) {
    if (!event.target.closest(".card")) {
      cards.forEach((card) => card.classList.remove("active")); // Retirer la classe active si on touche en dehors d'une carte
    }
  });
});
```

- **Logique de l'application** :

  - **Validation des formulaires** : Vérifiez les entrées utilisateur avant soumission.

```
$identifiant = $_POST['identifiant'];
$motdepasse = $_POST['motdepasse'];
```

4. PHP : Gestion côté serveur

Fichier PHP utilisés : connexion.php, global.php, sql.php, config.php et logout.php

- **Fonctionnalité principale :**

- **Gestion des connexions :** Le fichier connexion.php est responsable de l'authentification des utilisateurs du site, qu'ils soient administrateurs, employés ou vétérinaires. Il établit une connexion à la base de données, vérifie les identifiants des utilisateurs, et les redirige vers la page appropriée en fonction de leur rôle.

Structure du fichier connexion.php :

- **Connexion à la base de données :**

  - **Établissement de la connexion :** Utilise MySQLi pour créer une connexion à la base de données avec les informations définies dans config.php. La connexion est vérifiée pour s'assurer qu'elle est réussie.

```php
<?php

require_once('libs/global.php');

// Créer une connexion
$conn = new mysqli(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);

// Vérifier la connexion
if ($conn->connect_error) {
    die("Connexion échouée: " . $conn->connect_error);
}
```

- **Récupération et traitement des données du formulaire :**

  - **Extraction des données :** Les identifiants (nom d'utilisateur et mot de passe) sont récupérés à partir des données envoyées par le formulaire via la méthode POST.

  - **Préparation et exécution de la requête SQL :** Une requête SQL préparée est utilisée pour éviter les injections SQL. La requête vérifie si l'identifiant existe dans la base de données.

```php
// Récupérer les informations du formulaire
$identifiant = $_POST['identifiant'];
$motdepasse = $_POST['motdepasse'];
```

```
// Préparer et exécuter la requête SQL
$sql = "SELECT user_id, password, role FROM users WHERE username = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("s", $identifiant);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows < 1) {
    $_SESSION['error'] = 'Identifiant ou mot de passe incorrect';
    header('Location: /zooarcadia/connexion.html');
}

$user = $result→fetch_assoc();
```

- **Validation des identifiants :**

  - **Vérification du mot de passe :** Le mot de passe fourni est comparé avec le mot de passe haché stocké dans la base de données en utilisant password_verify(). Si la vérification échoue, l'utilisateur est redirigé vers la page de connexion avec un message d'erreur.

```
// Fermer la connexion
$conn->close();

// Vérifier le mot de passe
if (!password_verify($motdepasse, $user['password'])) {
    $_SESSION['error'] = 'Identifiant ou mot de passe incorrect';
    header('Location: /connexion.html');
}
```

- **Authentification réussie :**

  - **Gestion des sessions :** Si l'authentification est réussie, les informations de session sont définies (user_id et role). L'utilisateur est ensuite redirigé vers la page appropriée en fonction de son rôle.

```
// Authentification réussie
$_SESSION['user_id'] = $user['user_id'];
$_SESSION['role'] = $user['role'];

// Redirection basée sur le rôle
if ($user['role'] === 'veterinaire') {
    header('Location: /zooarcadia/connectveto.php');
} elseif ($user['role'] === 'admin') {
    header('Location: /zooarcadia/admin.php');
} elseif ($user['role'] === 'employe') {
```

```
    header('Location: /zooarcadia/employe.php');
} else {
    header('Location: /zooarcadia/connexion.php');
}
exit();
```

- **Gestion des erreurs :**
  - **Gestion des erreurs de connexion :** Si la connexion à la base de données échoue, un message d'erreur est affiché.
  - **Gestion des erreurs d'authentification :** Si les identifiants sont incorrects ou le mot de passe ne correspond pas, l'utilisateur est redirigé avec un message d'erreur.

Code complet de connexion.php :

```
connexion.php > ...
<?php

require_once('libs/global.php');

// Créer une connexion
$conn = new mysqli(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);

// Vérifier la connexion
if ($conn->connect_error) {
    die("Connexion échouée: " . $conn->connect_error);
}

// Récupérer les informations du formulaire
$identifiant = trim($_POST['identifiant']);
$motdepasse = trim($_POST['motdepasse']);

// Préparer et exécuter la requête SQL
$sql = "SELECT user_id, password, role FROM users WHERE username = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("s", $identifiant);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows < 1) {
    $_SESSION['error'] = 'Identifiant ou mot de passe incorrect';
    header('Location: /zooarcadia/connexions.php');
    exit(); // Ajout de exit() pour stopper l'exécution
}

$user = $result->fetch_assoc();
$stmt->close();
$conn->close();

// Vérifier le mot de passe
if (!password_verify($motdepasse, $user['password'])) {
    $_SESSION['error'] = 'Identifiant ou mot de passe incorrect';
    header('Location: /zooarcadia/connexions.php');
```

```php
        exit(); // Ajout de exit() pour stopper l'exécution
    }

    // Authentification réussie
    session_regenerate_id(true); // Sécuriser la session
    $_SESSION['user_id'] = $user['user_id'];
    $_SESSION['role'] = $user['role'];

    // Redirection basée sur le rôle
    if ($user['role'] === 'veterinaire') {
        header('Location: /zooarcadia/connectveto.php');
    } elseif ($user['role'] === 'admin') {
        header('Location: /zooarcadia/admin.php');
    } elseif ($user['role'] === 'employe') {
        header('Location: /zooarcadia/employe.php');
    } else {
        header('Location: /zooarcadia/connexions.php');
    }
    exit(); // Ajout de exit() pour stopper l'exécution

```

- **Intégration des fichiers complémentaires :**

  - **config.php :** Contient les informations de connexion à la base de données.

```
libs > config.php > ...
<?php

define('DB_HOST', 'localhost');
define('DB_NAME', 'arcadia1');
define('DB_USER', 'root');
define('DB_PASSWORD', 'root');
```

  - **sql.php :** Définit la fonction initConnexion() pour établir une connexion à la base de données.

```
libs > sql.php > ...
<?php

6 references
function initConnexion()
{
    // Créer une connexion
    $conn = new mysqli(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);
    $conn->set_charset("utf8mb4");

    // Vérifier la connexion
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    return $conn;
}
```

- **global.php :** Contient des fonctions globales et est inclus pour des opérations comme la vérification d'accès et la gestion des sessions.

```
global.php > PHP Intelephense > checkAccess
<?php

session_start();

require_once('libs/config.php');
require_once('libs/sql.php');

3 references
function checkAccess() {
    if (!isset($_SESSION['user_id'])) {
        header('Location: /connexions.php');
        die();
    }
}
```

Cette structure assure une gestion sécurisée et efficace des connexions des utilisateurs, en vérifiant les identifiants et en dirigeant les utilisateurs vers les pages appropriées selon leur rôle dans l'application.

- **logout.php** : Ce code PHP est utilisé pour déconnecter un utilisateur en supprimant toutes les variables de session et en détruisant complètement la session en cours. Ensuite, il redirige l'utilisateur vers la page d'accueil (`index.html`).

logout.php

```php
<?php

session_start();
session_unset();
session_destroy();

header('Location: index.html');

```

5. SQL : Base de données

Structure de la base de données :

1. **Table animal_data**

- **Champs :**
  - id (int) : Identifiant unique, clé primaire, AUTO_INCREMENT
  - animal (varchar) : Nom de l'animal
  - quantity (int) : Quantité associée à l'animal
  - checkup_date (date) : Date de vérification
  - health (text) : Évaluation de la santé
  - user_id (int) : Identifiant de l'utilisateur, clé étrangère

- **Clés primaires :**
  - id

- **Clés étrangères :**
  - user_id → users.user_i

```sql
CREATE TABLE `animal_data` (
  `id` int(11) NOT NULL,
  `animal` varchar(255) NOT NULL,
  `quantity` int(11) NOT NULL,
```

```
  `checkup_date` date NOT NULL,
  `health` text NOT NULL,
  `user_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

2. **Table avis**

- **Champs :**
  - id (int) : Identifiant unique, clé primaire, AUTO_INCREMENT
  - pseudo (varchar) : Nom du rédacteur de l'avis
  - texte (text) : Contenu de l'avis
  - date (timestamp) : Date et heure de l'avis
- **Clés primaires :**
  - id

```
CREATE TABLE `avis` (
  `id` int(11) NOT NULL,
  `pseudo` varchar(100) NOT NULL,
  `texte` text NOT NULL,
  `date` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

3. **Table habitat_clicks**
- **Champs :**
  - id (int) : Identifiant unique, clé primaire, AUTO_INCREMENT
  - habitat (varchar) : Nom de l'habitat
  - clicks (int) : Nombre de clics
- **Clés primaires :**
  - id
- **Index unique :**
  - habitat

```
CREATE TABLE `habitat_clicks` (
  `id` int(11) NOT NULL,
  `habitat` varchar(255) NOT NULL,
  `clicks` int(11) DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

4. **Table users**
    - **Champs :**
        - user_id (int) : Identifiant unique, clé primaire, AUTO_INCREMENT
        - username (varchar) : Nom d'utilisateur, unique
        - password (varchar) : Mot de passe (haché)
        - role (enum) : Rôle (admin, employe, veterinaire)
    - **Clés primaires :**
        - user_id
    - **Index unique :**
        - username

Clés primaires et étrangères :

- **Clés primaires :**
    - animal_data.id
    - avis.id
    - habitat_clicks.id
    - users.user_id
- **Clés étrangères :**
    - animal_data.user_id → users.user_id

```
CREATE TABLE `users` (
  `user_id` int(11) NOT NULL,
  `username` varchar(50) NOT NULL,
  `password` varchar(255) NOT NULL,
  `role` enum('admin','employe','veterinaire') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
INSERT INTO `users` (`user_id`, `username`, `password`, `role`) VALUES
(1, 'admin', 'adminpass', 'admin'),
```

```
(2, 'employe', '$2y$10$examplehash2', 'employe'),
(3, 'veterinaire', '$2y$10$examplehash3', 'veterinaire');
```

- **AUTO_INCREMENT :**
    - Activé pour id dans animal_data, avis, habitat_clicks et user_id dans users

Requêtes SQL :

- **CRUD (Create, Read, Update, Delete) :**
    - **SELECT** : Pour afficher les données (ex. : consulter les avis ou horaires des repas)
    - **INSERT** : Pour ajouter de nouvelles données (ex. : ajouter un nouvel avis ou une nouvelle entrée de repas)
    - **UPDATE** : Pour modifier des informations existantes (ex. : mettre à jour les détails d'un animal)
    - **DELETE** : Pour supprimer des données (ex. : supprimer un avis ou une entrée de repas)

6. NoSQL : Base de données pour les clics réseaux sociaux

Ce script JavaScript utilise Firebase pour enregistrer les clics sur des liens de réseaux sociaux dans une base de données Firestore.

1. **Importation des modules** : Il importe les fonctions nécessaires depuis les bibliothèques Firebase pour initialiser l'application et accéder à Firestore.

2. **Configuration de Firebase** : Il configure Firebase avec les informations de projet, telles que la clé API et l'identifiant du projet.

3. **Initialisation de Firebase** : Le script initialise Firebase avec la configuration fournie et crée une instance de Firestore pour interagir avec la base de données.

4. **Enregistrement des clics** : Lorsque le DOM est entièrement chargé, le script ajoute un gestionnaire d'événements à tous les liens dans les éléments de la classe .formcontact. Lorsqu'un lien est cliqué, il vérifie si l'URL du lien contient 'instagram' ou 'facebook' pour

déterminer la plateforme, puis enregistre cette information, ainsi que l'heure du clic, dans Firestore.

5. **Gestion des erreurs** : Il enregistre un message dans la console en cas d'erreur lors de l'ajout du document dans la base de données

```
<script type="module">
  import { initializeApp } from "https://www.gstatic.com/firebasejs/9.9.4/firebase-app.js";
  import { getFirestore, collection, addDoc } from
"https://www.gstatic.com/firebasejs/9.9.4/firebase-firestore.js";

  const firebaseConfig = {
    apiKey: "AIzaSyAxJ5mKB2DEzRAi8NoVfEG89wM3Qpab35Q",
    authDomain: "arcadia14.firebaseapp.com",
    projectId: "arcadia14",
    storageBucket: "arcadia14.appspot.com",
    messagingSenderId: "230094407275",
    appId: "1:230094407275:web:65ea5786845081f98dd32e",
    measurementId: "G-5479SQWV6V"
  };

  // Initialize Firebase
  const app = initializeApp(firebaseConfig);
  const db = getFirestore(app);

  document.addEventListener('DOMContentLoaded', () => {
    const logClick = async (platform) => {
      try {
        await addDoc(collection(db, 'clicks'), {
          platform: platform,
          timestamp: new Date().toISOString(),
        });
        console.log(`Click recorded: ${platform}`);
      } catch (error) {
        console.error("Error adding document: ", error);
      }
    };

    document.querySelectorAll('.formcontact a').forEach(link => {
      link.addEventListener('click', (event) => {
        const platform = event.currentTarget.href.includes('instagram') ? 'Instagram' :
'Facebook';
        logClick(platform);
      });
    });
  });
</script>
```

Styles et Design

Charte Graphique pour Le Zoo Arcadia

Polices Utilisées

1. **Police principale : Geneva, Tahoma, sans-serif**

    - **Description** : Ces polices sans-serif classiques sont utilisées pour offrir une apparence propre, moderne et facile à lire sur l'ensemble du site. Elles sont uniformément appliquées dans les titres, paragraphes et boutons.

Palette de Couleurs

1. Couleurs principales

- **Vert foncé** : #445843
    - *Utilisation* : Fond des éléments de navigation, titres, et certaines sections de contenu. Ce vert profond contraste bien avec les autres éléments pour guider l'attention des utilisateurs.

- **Gris foncé** : #444444
    - *Utilisation* : Fond des titres dans certaines sections. Il est utilisé pour donner du contraste et de la profondeur.

Dégradés :

- **Dégradé pour les sections entre les headers et footers** : linear-gradient(to bottom, #445843, #455e3b)

2. Couleurs des textes

- **Blanc cassé** : #F0F0FF
  - *Utilisation* : Texte principal sur les fonds colorés, offrant un contraste doux et une bonne lisibilité.

- **Noir** : #000000
  - *Utilisation* : Texte principal sur les fonds clairs, notamment dans les sections mentions légales et RGPD.

- **Gris clair** : #D3D3D3
  - *Utilisation* : Texte des jours ou informations secondaires.

- **Blanc** : #FFFFFF
  - *Utilisation* : Texte principal dans les sections avec un fond plus sombre, par exemple sur les boutons ou en-têtes.

3. Autres couleurs

- **Vert clair** : lightgreen
  - *Utilisation* : Texte du titre section « Nos Horaires ».

- **Gris clair** : lightgrey
  - *Utilisation* : Texte des jours section « Nos Horaires ».

4. Effets et ombres

- **Ombres portées sur les éléments** :

- *Boutons et cartes* : Les boutons et certaines cartes de contenu sur le site bénéficient d'une ombre portée subtile (box-shadow) pour leur donner un effet de relief. Cela rend les éléments interactifs plus visibles et engageants.

- *Titres et sections spécifiques* : Certains titres et sections utilisent également des ombres portées légères pour les faire ressortir, surtout sur les fonds plus clairs.

5. Fond du site web

- **Image de fond** : Une image d'arrière-plan représentant un paysage naturel avec une végétation luxuriante, probablement une forêt ou une savane avec des arbres, compatible avec le thème animalier.

    - *Utilisation* : Cette image est utilisée comme fond général du site, ajoutant de la profondeur et une atmosphère immersive qui complète le thème du parc zoologique. L'image est subtile, permettant au contenu principal de rester lisible et de se détacher du fond.

Observations supplémentaires :

- **Structure visuelle** : Le site présente une mise en page simple et claire, facilitant la navigation. La palette de couleurs naturelles et la typographie contribuent à une ambiance accueillante et familière, idéale pour un site dédié à un parc animalier.

- **Consistance des styles** : La cohérence dans l'utilisation des couleurs et des polices est bien respectée, créant une expérience utilisateur harmonieuse.

- **Interactivité** : Les couleurs de survol (hover) sont bien utilisées pour indiquer l'interaction avec les éléments cliquables, rendant l'interface utilisateur plus intuitive.

- **Image de fond** : L'image de fond joue un rôle clé dans l'atmosphère générale du site, renforçant le lien avec la nature et les animaux. Elle est choisie pour sa capacité à améliorer l'esthétique sans nuire à la lisibilité du contenu.

Maintenance et Support

1. **Surveillance**: Utiliser des outils de surveillance pour suivre les performances et les erreurs.

2. **Mises à Jour**: Mettre à jour régulièrement les dépendances et le code pour corriger les bugs et ajouter de nouvelles fonctionnalités.

3. Sécurité des Formulaires

1. **Protection contre les Attaques de Type Cross-Site Scripting (XSS) :**
   - **Échappement des Données Utilisateur :** Lors de l'affichage de données utilisateur sur le site, toutes les entrées sont échappées pour prévenir les injections de scripts malveillants. Ceci est réalisé en utilisant des fonctions PHP telles que htmlspecialchars().

Ce code de la page **admin.php** prépare et exécute une requête d'insertion dans la base de données pour ajouter des informations sur des soins d'animaux. Il vérifie d'abord si la préparation de la requête SQL a réussi, puis lie les paramètres (animal_id, quantity, checkup_date, health, et user_id) à la requête. Si l'exécution de la requête échoue, une erreur est affichée.

htmlspecialchars() est utilisée ici pour **échapper les erreurs SQL** et les afficher de manière sécurisée, empêchant ainsi l'injection de code HTML ou de scripts dans le message d'erreur affiché.

```
        // Préparer et exécuter la requête d'insertion
    $stmt = $conn->prepare("INSERT INTO animal_data (animal, quantity, checkup_date, health, user_id) VALUES (?, ?, ?, ?, ?)");
    if ($stmt === false) {
        die("Prepare failed: " . htmlspecialchars($conn->error, ENT_QUOTES, 'UTF-8'));
    }

    $stmt->bind_param('sissi', $animal_id, $quantity, $checkup_date, $health, $_SESSION['user_id']);
    if (!$stmt->execute()) {
        die("Execute failed: " . htmlspecialchars($stmt->error, ENT_QUOTES, 'UTF-8'));
    }
```

2. **Protection contre les Injections SQL :**

- **Requêtes Préparées :** Les interactions avec la base de données utilisent des requêtes préparées avec des paramètres liés pour éviter les injections SQL. Par exemple, lors de la connexion, les identifiants de l'utilisateur sont passés via des paramètres liés dans les requêtes préparées.

Ce code présent dans **connexion.php** prépare et exécute une requête SQL pour vérifier les informations de connexion d'un utilisateur. Il sélectionne le user_id, le password haché, et le role de la table users où le nom d'utilisateur correspond à celui fourni ($identifiant). Si aucun utilisateur n'est trouvé, un message d'erreur est stocké dans la session et l'utilisateur est redirigé. Ensuite, les informations de l'utilisateur sont récupérées avec fetch_assoc() et la connexion à la base de données est fermée.

```
        // Préparer et exécuter la requête SQL
$sql = "SELECT user_id, password, role FROM users WHERE username = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("s", $identifiant);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows < 1) {
    $_SESSION['error'] = 'Identifiant ou mot de passe incorrect';
    header('Location: /zooarcadia/connexion.html');
}

$user = $result->fetch_assoc();
// Fermer la connexion
$conn->close();
```

3. **Protection des Données Sensibles :**

- **Hashage des Mots de Passe :** Les mots de passe sont stockés dans la base de données sous forme de hachage sécurisé en utilisant des algorithmes modernes comme bcrypt, garantissant que les mots de passe ne sont pas stockés en clair.

- **Gestion des Sessions :** Les sessions utilisateur sont protégées contre les détournements en utilisant des identifiants de session uniques et en régénérant les identifiants de session après une connexion réussie.

```
        // Authentification réussie
session_regenerate_id(true); // Sécuriser la session
```

```
$_SESSION['user_id'] = $user['user_id'];
$_SESSION['role'] = $user['role'];
```

Application des Mesures dans le Code

Voici comment ces mesures sont appliquées dans le code PHP fourni :

- **Requêtes Préparées :** Utilisation de requêtes préparées ($stmt = $conn->prepare($sql)) pour les interactions avec la base de données afin d'éviter les injections SQL.

- **Hashage des Mots de Passe :** Utilisation de password_verify() pour comparer les mots de passe lors de la connexion.

- **Gestion des Sessions :** Stockage sécurisé des informations de session et régénération de l'identifiant de session après la connexion.

Fonctionnalités de Sécurité de Session et Authentification

1. **Démarrage de la Session** :
    - session_start(); est appelé au début du script pour initialiser la gestion des sessions.

2. **Trim des Données d'Entrée** :
    - Les données du formulaire sont nettoyées avec trim() pour enlever les espaces superflus autour des entrées.

3. **Préparation et Exécution de la Requête SQL** :
    - La requête SQL est préparée et les paramètres sont liés pour éviter les injections SQL.

4. **Vérification de l'Existence de l'Utilisateur** :
    - Si aucun utilisateur n'est trouvé ($result->num_rows < 1), une erreur est stockée dans $_SESSION['error'], et l'utilisateur est redirigé vers la page de connexion.

5. **Vérification du Mot de Passe** :
    - password_verify($motdepasse, $user['password']) est utilisé pour vérifier que le mot de passe fourni correspond au mot de passe haché stocké en base de données. En cas d'échec, une erreur est enregistrée dans la session et l'utilisateur est redirigé.

6. **Régénération de l'ID de Session** :
    - session_regenerate_id(true); est utilisé après une connexion réussie pour empêcher les détournements de session. Cela génère un nouvel identifiant de session, ce qui renforce la sécurité.

7. **Stockage des Informations de Session** :
    - Les informations utilisateur (user_id et role) sont stockées dans la session pour permettre une gestion des rôles et une personnalisation basée sur l'utilisateur connecté.

8. **Redirection Basée sur le Rôle** :
    - Selon le rôle de l'utilisateur (veterinaire, admin, ou employe), il est redirigé vers la page appropriée. Si aucun rôle ne correspond, l'utilisateur est renvoyé à la page de connexion.

9. **Utilisation de exit()** :
    - exit(); est utilisé après chaque redirection pour s'assurer que le script s'arrête immédiatement et évite tout traitement supplémentaire.

Ces mesures combinées assurent que les formulaires sont protégés contre les saisies malveillantes, les données utilisateur sont traitées de manière sécurisée, et les interactions avec la base de données sont sécurisées contre les injections SQL et autres attaques.

Bonnes Pratiques

- **Code Propre**: Écrire du code propre et bien commenté.
- **Accessibilité**: Site est accessible à tous les utilisateurs.
- **Performances**: Optimisez les performances du site en minimisant les ressources.

Ressources et Références

- MDN Web Docs
- GitHub Documentation

Documentation Technique du Projet "Le Zoo Arcadia"

Introduction

Ce document technique détaille les technologies utilisées pour le développement du projet "Le Zoo Arcadia", ainsi que la configuration et le déploiement sur O2Switch. Le projet utilise Visual Studio Code (VS Code) comme éditeur de code, et les technologies web standards HTML, CSS et JavaScript , PHP et MySQL pour le développement du site web.

Réflexion Initiale du Projet "Le Zoo Arcadia"

Contexte et Objectifs

Le projet "Le Zoo Arcadia" est né de l'idée de créer une plateforme en ligne pour un zoo fictif, Arcadia, où les visiteurs peuvent obtenir des informations sur les animaux, les attractions et les services offerts. L'objectif principal est de fournir une expérience utilisateur engageante et informative qui incite les visiteurs à planifier leur visite au zoo.

Public Cible

Le site web vise principalement les familles, les enseignants, et les amateurs d'animaux qui cherchent des informations sur les activités éducatives et récréatives du zoo. Il s'adresse aussi aux membres de la communauté locale et aux touristes intéressés par une visite.

Fonctionnalités Clés

1. **Présentation des Animaux** : Informations détaillées sur les animaux, y compris leur habitat, leur alimentation, et des faits intéressants. Cette fonctionnalité permet aux visiteurs de découvrir les différentes espèces présentes dans le zoo et d'en apprendre davantage sur chacune d'elles.

2. **Plan du Zoo** : Carte interactive du zoo pour aider les visiteurs à planifier leur parcours. La carte permet de localiser les différents enclos, services, et points d'intérêt, facilitant ainsi la navigation à travers le zoo.

3. **Informations Pratiques** : Horaires d'ouverture, services disponibles (restaurants, circuit de petit train, etc.). Cette section fournit des informations essentielles pour la planification de la visite, incluant les horaires de fonctionnement du zoo et les services offerts aux visiteurs.

4. **Connexion Utilisateur** : Système de connexion pour les différents types d'utilisateurs (administrateurs, employés, vétérinaires). Les utilisateurs peuvent se connecter via un formulaire sécurisé pour accéder à des fonctionnalités spécifiques, telles que la gestion des suivis sur les animaux, la modification de leur état de santé, la mise à jour des repas et même le suivi des clics sur les différents habitats.

Défis Anticipés et contraintes

1. **Compatibilité et Accessibilité** : Assurer que le site fonctionne sur tous les navigateurs modernes et est accessible facilement.

2. **Performance** : Optimiser le site pour qu'il charge rapidement et fonctionne sans accroc, même avec un grand nombre de visiteurs.

3. **Contenu Interactif** : Créer une expérience utilisateur engageante avec des éléments interactifs sans sacrifier la performance ou la compatibilité.

Technologies Utilisées

Visual Studio Code (VS Code)

VS Code est un éditeur de code source développé par Microsoft. Il est léger, puissant et dispose de nombreuses extensions qui facilitent le développement web.

- **Léger et rapide** : VS Code est conçu pour être léger et rapide, permettant un développement fluide même sur des machines moins puissantes.
- **Extensions** : Une vaste bibliothèque d'extensions pour augmenter la productivité (ex. : Prettier pour le formatage de code, Live Server pour le développement en temps réel).
- **Intégration Git** : Intégration native avec Git, facilitant le contrôle de version.
- **Github :** Pour l'hébergement des données.

HTML, CSS, JavaScript, PHP et MySQL

Ces cinq technologies sont essentielles pour le développement web complet et interactif.

- **HTML (HyperText Markup Language)** : Utilisé pour structurer le contenu du site web. HTML définit la structure de base des pages web en utilisant des balises pour organiser le texte, les images, les liens, et autres éléments.

- **CSS (Cascading Style Sheets)** : Utilisé pour styliser le contenu HTML, en améliorant l'apparence visuelle du site. CSS permet de contrôler la mise en page, les couleurs, les polices, et les effets visuels, assurant ainsi une présentation cohérente et attrayante.

- **JavaScript** :
- **Sur Arcadia** : Pour ajouter des fonctionnalités interactives, comme la carte interactive du zoo et les animations dynamiques.

- **PHP (Hypertext Preprocessor)** :
- **Sur Arcadia** : Pour entre autres gérer les connexions des utilisateurs (administrateurs, employés, vétérinaires).

  connexion.php est utilisé pour authentifier les utilisateurs et gérer les sessions.

  config.php contient les informations de configuration de la base de données, telles que l'hôte, l'utilisateur, le mot de passe et le nom de la base de données.

  sql.php contient une fonction pour établir une connexion à la base de données en utilisant MySQLi.

- **MySQL** :
- **Sur Arcadia**: Pour la gestion des données relatives aux animaux, aux avis, aux habitats, aux soins, aux repas et aux utilisateurs. La base de données arcadia utilise des tables pour stocker et organiser les informations nécessaires au bon fonctionnement du site, avec des relations entre les tables pour assurer l'intégrité des données.

Développement et Déploiement

Développement Local

1. **Lancer le Serveur Local** :

   Instructions de Déploiement en local

Prérequis XAMPP : Assurez-vous que XAMPP est installé sur votre machine. XAMPP inclut Apache, MySQL et PHP, qui sont nécessaires pour exécuter des sites web dynamiques en local.

VSCode : Utilisez un éditeur de code comme Visual Studio Code pour modifier et gérer votre code.

phpMyAdmin : Pour gérer vos bases de données MySQL, phpMyAdmin est intégré dans XAMPP, vous permettant de manipuler et administrer vos bases de données facilement via une interface web.

1.Installer XAMPP et Configurer l'Environnement Lancer XAMPP: Ouvrez XAMPP et démarrez les modules Apache et MySQL. Répertoire de Travail: Placez vos fichiers dans le dossier htdocs de XAMPP (généralement situé dans C:/xampp/htdocs).

2.Cloner le Repository (facultatif) Si vous souhaitez travailler sur votre projet localement, suivez les étapes ci-dessous pour cloner votre repository depuis GitHub et l'installer dans le répertoire XAMPP.

Étapes : Ouvrez VSCode et ouvrez un terminal intégré.

Exécutez la commande suivante pour cloner le repository :

git clone https://github.com/Bendbds/le-zoo-arcadia.git Accédez au répertoire du projet

cd le-zoo-arcadia Déplacez le dossier cloné dans C:/xampp/htdocs/.

3.Configurer la Base de Données via phpMyAdmin

Si votre site utilise une base de données, vous devrez la configurer via phpMyAdmin.

Étapes : Ouvrez un navigateur et allez à http://localhost/phpmyadmin. Créez une nouvelle base de données pour votre projet. Importez un fichier SQL (si vous en avez un) en utilisant l'onglet Import. Modifiez les fichiers de configuration de votre projet (comme config.php) pour pointer vers cette nouvelle base de données, par exemple :

```
php
define('DB_HOST', 'localhost');
 define('DB_USER', 'root'); // L'utilisateur par défaut
define('DB_PASSWORD', ''); // Pas de mot de passe par défaut
define('DB_NAME', 'nom_de_votre_base_de_données');
```

4.Accéder à Votre Projet Localement Ouvrez un navigateur et allez à http://localhost/le-zoo-arcadia/ pour voir votre projet en local.

Déploiement sur O2SWITCH

1. Connexion à cPanel pour O2Switch Connectez-vous à votre compte o2switch et accédez à cPanel.

Dans cPanel, allez dans Gestion des fichiers et ouvrez le répertoire public_html ou le répertoire ou vous comptez installer votre site web.

Transférer les Fichiers vers le Serveur Utilisez un client FTP comme FileZilla ou le gestionnaire de fichiers o2Switch pour vous connecter à votre serveur o2switch.

Les informations FTP sont disponibles dans votre cPanel, sous Comptes FTP.

Naviguez dans le répertoire public_html. Si vous souhaitez déployer le site à la racine de votre domaine principal, c'est ici que vous placerez vos fichiers.

Téléversez les fichiers du projet (ou le contenu du dossier build/dist si vous avez effectué un build) dans ce répertoire.

Configuration du Domaine (si nécessaire) Si votre domaine n'est pas encore configuré, assurez-vous de pointer les enregistrements DNS de votre domaine vers les serveurs de noms d'o2switch. Ces informations se trouvent dans le panneau d'administration de votre domaine.

Si vous souhaitez déployer le site dans un sous-dossier ou sur un sous-domaine, configurez cela via cPanel en utilisant l'option Sous-domaines ou Domaines supplémentaires.

Tester le Déploiement Une fois les fichiers téléversés, ouvrez votre navigateur et allez à votre domaine ou sous-domaine pour vérifier que le site "Le Zoo d'Arcadia" est bien en ligne.

Vérifiez que toutes les pages se chargent correctement et que le site fonctionne comme prévu.

Configurer un Certificat SSL (HTTPS) Dans cPanel, allez dans la section Let's Encrypt™ SSL.

Sélectionnez votre domaine et suivez les instructions pour générer un certificat SSL gratuit.

Une fois le certificat installé, assurez-vous que toutes les URL de votre site utilisent HTTPS pour une sécurité accrue.

Notes Additionnelles Si vous rencontrez des erreurs 500 ou d'autres problèmes liés au serveur, vérifiez les fichiers .htaccess pour toute mauvaise configuration. Voilà, le site "Le Zoo d'Arcadia" est maintenant être en ligne et accessible sur o2switch !

Conclusion

Le projet "Le Zoo Arcadia" a pour objectif de créer une plateforme en ligne attrayante et informative pour les visiteurs du zoo. En utilisant HTML pour structurer le contenu de manière sémantique, CSS pour appliquer une mise en forme moderne et responsive, et JavaScript pour ajouter des fonctionnalités interactives telles que des cartes et des animations dynamiques, je vise à offrir une expérience utilisateur enrichissante et engageante.

À ce jour, la majorité des fonctionnalités demandées pour le site web d'Arcadia sont mises en place avec un respect rigoureux des normes de sécurité. Les fonctionnalités essentielles, telles que la gestion des avis et des commentaires, ainsi que les outils de suivi des animaux, ont été développées et sécurisées avec soin. Toutefois, certaines fonctionnalités prévues, comme le filtrage des véhicules pour l'exposition, ont été reportées pour se concentrer sur des aspects jugés plus critiques pour le bon fonctionnement du site.

À l’avenir, plusieurs axes d’amélioration sont envisagés pour enrichir l’expérience utilisateur. Parmi ces améliorations, la création d’un plan téléchargeable du zoo pourrait faciliter la visite des visiteurs, et un contrôle renforcé des avis permettrait de mieux gérer les retours et les commentaires. De plus, une amélioration dans la saisie des états des animaux pourrait offrir une plus grande flexibilité pour les vétérinaires et les employés, facilitant ainsi leur travail quotidien.

Le développement de ce site a représenté un véritable défi. Les connaissances acquises au cours de ma formation ont été complétées par des recherches approfondies à travers divers supports, tels que forums, documentations, et vidéos tutoriels. Cette démarche m’a permis de mieux comprendre certaines notions sous différents angles et d'enrichir ma boîte à outils avec de nouvelles techniques et méthodes. Grâce à cette approche, j'ai non seulement renforcé mes compétences techniques, mais aussi développé des compétences transversales cruciales pour le métier de développeur web.

Conscient que le métier est en perpétuelle évolution, je suis déterminé à continuer à renforcer ces compétences tout au long de ma carrière, en restant à l’affût des innovations et en proposant des solutions toujours plus pertinentes et novatrices.

Annexes

Code complet de la page admin.php

Ce code PHP gère l'administration du zoo Arcadia en permettant l'ajout et la gestion des données sur les soins des animaux ainsi que le suivi des clics des habitats. Il inclut des fonctions pour se connecter à la base de données, insérer des données de soins, réinitialiser les clics des habitats, et récupérer les données pour affichage. Le code HTML affiche un formulaire pour saisir les soins des animaux, une liste des clics des habitats, et un affichage des soins enregistrés, avec des messages de confirmation et des boutons pour réinitialiser les clics.

```php
<?php
require_once('libs/global.php');

checkAccess();

$success = false;

// Initialiser la connexion à la base de données
$conn = initConnexion();

// Traitement du formulaire d'ajout de données de soins des animaux
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['animal_id'], $_POST['quantity'], $_POST['checkup_date'], $_POST['health'])) {
    $animal_id = trim($_POST['animal_id']);
    $quantity = (int)$_POST['quantity']; // Assurez-vous que la quantité est un entier
    $checkup_date = $_POST['checkup_date'];
    $health = trim($_POST['health']);

    // Préparer et exécuter la requête d'insertion
    $stmt = $conn->prepare("INSERT INTO animal_data (animal, quantity, checkup_date, health, user_id) VALUES (?, ?, ?, ?, ?)");
    if ($stmt === false) {
        die("Prepare failed: " . htmlspecialchars($conn->error, ENT_QUOTES, 'UTF-8'));
    }

    $stmt->bind_param('sissi', $animal_id, $quantity, $checkup_date, $health, $_SESSION['user_id']);
    if (!$stmt->execute()) {
        die("Execute failed: " . htmlspecialchars($stmt->error, ENT_QUOTES, 'UTF-8'));
    }

    $stmt->close();

    // Stocker la réussite dans une variable de session
    $_SESSION['success'] = true;

    // Redirige l'utilisateur vers la même page avec une requête GET
    header("Location: " . htmlspecialchars($_SERVER['PHP_SELF'], ENT_QUOTES, 'UTF-8'));
    exit();
```

```php
}

// Traitement du formulaire pour réinitialiser les clics
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['reset'])) {
    $stmt = $conn->prepare("UPDATE habitat_clicks SET clicks = 0");
    if ($stmt === false) {
        die("Prepare failed: " . htmlspecialchars($conn->error, ENT_QUOTES, 'UTF-8'));
    }

    if (!$stmt->execute()) {
        die("Execute failed: " . htmlspecialchars($stmt->error, ENT_QUOTES, 'UTF-8'));
    }

    $stmt->close();

    // Rediriger vers la même page pour rafraîchir les données
    header("Location: " . htmlspecialchars($_SERVER['PHP_SELF'], ENT_QUOTES, 'UTF-8'));
    exit();
}

// Préparer et exécuter la requête de sélection pour les clics des habitats
$habitatClicksQuery = "SELECT * FROM habitat_clicks";
$clicksStmt = $conn->prepare($habitatClicksQuery);
if ($clicksStmt === false) {
    die("Prepare failed: " . htmlspecialchars($conn->error, ENT_QUOTES, 'UTF-8'));
}

$clicksStmt->execute();
$clicksResult = $clicksStmt->get_result();
$clicksData = $clicksResult->fetch_all(MYSQLI_ASSOC);
$clicksStmt->close();

// Préparer et exécuter la requête de sélection pour les données de soins des animaux
$sql = "SELECT * FROM animal_data a
        JOIN users u ON u.user_id = a.user_id
        ORDER BY checkup_date DESC";
```

```php
$stmt = $conn->prepare($sql);
if ($stmt === false) {
    die("Prepare failed: " . htmlspecialchars($conn->error, ENT_QUOTES, 'UTF-8'));
}

$stmt->execute();
$result = $stmt->get_result();
$stmt->close();

$conn->close();
?>
```

Wireframe site zoo Arcadia

Le zoo d'Arcadia est un zoo à vocation ecoresponsable, le respect des animaux et de leurs habitats naturels est une priorité. Entièrement indépendant en énergie, nous sommes fiers de nos valeurs écologiques.

Le zoo met tout en œuvre pour développer et soutenir des actions de conservation pour la préservation de la biodiversité mondiale. Son objectif est de sensibiliser le grand public et les visiteurs du ZooParc Arcadia en particulier à la nécessité de ces actions.

Plus de 150 animaux sauvages vivent paisiblement en semi-liberté sur 80 hectares de forêts, plaines et vallons. Tigres, lions, pandas, girafes, partez à la découverte de 120 espèces des 5 continents à travers 2 circuits de visite à pied et observez les animaux dans leurs habitats naturels. Ici, tout a été conçu pour le bien-être des animaux. Le parc propose également plusieurs attractions pour petits et grands. Une aventure exceptionnelle au coeur de Brocéliande à vivre en famille ou entre amis.

Nos horaires

Lundi	08:00 - 18:00
Mardi	08:00 - 18:00
Mercredi	08:00 - 18:00
Jeudi	08:00 - 18:00
Vendredi	08:00 - 18:00
Samedi	08:00 - 18:00
Dimanche	08:00 - 18:00

Votre avis compte pour nous!

Martine C :

L'accueil au top ! Les activités pédagogiques au top ! L'organisation est bien gérée. Le tarif vaut la sortie. Je vous recommande!

Vanessa R :

Très beau parc ombragé, animaux magnifiques, de beaux espaces, le nécessaire pour se restaurer, très chouette visite en famille !

Mickael T :

Une visite des plus merveilleuses comme à chaque fois ! Les vastes enclos, les paysages, les animaux qui semblent être heureux ! Un régal ! Sans compter les nouveaux aménagements qui sont très bien pensés ! Bravo !

Visite en petit train

Restauration sur place

arcadia zoo

Menu

Plus de 150 animaux sauvages vivent paisiblement en semi-liberté sur 80 hectares de forêts, plaines et vallons. Tigres, lions, pandas, girafes, partez à la découverte de 120 espèces des 5 continents à travers 2 circuits de visite à pied et observez les animaux dans leurs habitats naturels. Ici, tout a été conçu pour le bien-être des animaux. Le parc propose également plusieurs attractions pour petits et grands. Une aventure exceptionnelle au coeur de Brocéliande à vivre en famille ou entre amis.

Nos horaires

Lundi	08:00 - 18:00
Mardi	08:00 - 18:00
Mercredi	08:00 - 18:00
Jeudi	08:00 - 18:00
Vendredi	08:00 - 18:00
Samedi	08:00 - 18:00
Dimanche	08:00 - 18:00

Votre avis compte pour nous!

Martine C :

L'accueil au top ! Les activités pédagogiques au top ! L'organisation est bien gérée. Le tarif vaut la sortie. Je vous recommande!

Vanessa R :

Très beau parc ombragé, animaux magnifiques, de beaux espaces, le nécessaire pour se restaurer, très chouette visite en famille !

Mickael T :

Une visite des plus merveilleuses comme à chaque fois ! Les vastes enclos, les paysages, les animaux qui semblent être heureux ! Un régal ! Sans compter les nouveaux aménagements qui sont très bien pensés ! Bravo !

Visite en petit train

Restauration sur place

Mentions légales

Politique de données personnelles

Votre zoo Arcadia vous propose différents services afin de vous offrir une experience exceptionnelle:

Au menu du restaurant

Les petits plus de votre zoo

arcadia zoo

Menu

Votre zoo Arcadia vous propose différents services afin de vous offrir une experience exceptionnelle:

Au menu du restaurant

Les petits plus de votre zoo

Mentions légales

Politique de données personnelles

Pour nous contacter

Instag
ram

Faceb
ook

Par mail: arcadia.broceliande@gmail.com

Menu

Pour nous contacter

Instagram

Facebook

Par mail:
arcadia.broceliande@gmail.com

Mentions légales

Politique de données personnelles

Manuel d'Utilisation du Site Web "Le Zoo d'Arcadia"

Introduction

Bienvenue sur le site web du Zoo d'Arcadia ! Ce manuel d'utilisation vous guidera à travers les fonctionnalités et les sections principales du site, afin de vous offrir une expérience de navigation optimale.

Accéder au Site

Pour visiter notre site web, entrez l'URL suivante dans votre navigateur web : **https://benoit-dbds.fr/**

Note : Le site est hébergé sur o2switch, un service d'hébergement fiable offrant une performance et une sécurité accrues pour une navigation fluide et sécurisée.

Page d'Accueil

La page d'accueil offre un aperçu général du zoo, incluant :

- Un diaporama de photos mettant en valeur les animaux et installations.
- Des liens rapides vers les sections principales du site.
- Les horaires d'ouverture.
- Vos exemples d’avis sur le zoo.
- Mentions légales et conformité RGPD.

Navigation Principale

La barre de navigation en haut de chaque page vous permet d'accéder rapidement aux différentes sections du site :

- **Accueil** : Retournez à la page d'accueil depuis n'importe quelle page du site en cliquant sur le logo.
- **Habitats** : Découvrez un échantillon des différentes espèces présentes au zoo dans leurs habitats naturels.
- **Services** : Informations pratiques pour planifier votre visite : Menu du restaurant, plan du zoo, petit train et aire de jeux.
- **Connexion employés** : Formulaire de connexion avec droits spécifiques en fonction du poste : employé / vétérinaire / administrateur.
- **Contact** : Page de contact pour nous joindre et naviguer sur les réseaux sociaux.
- **Avis** : Formulaire de dépôt d’avis client sur le zoo.

Sections Détaillées

Accueil

- **Texte d'Introduction** : Une brève introduction au Zoo d'Arcadia, soulignant notre mission et nos valeurs.
- **Explication du Fonctionnement et Valeurs du Zoo** : Découvrez comment nous fonctionnons et ce qui nous distingue.

- **Horaires Détaillés** : Consultez les horaires d'ouverture du zoo.

- **Exemples d’Avis sur le Zoo** : Lisez les avis de nos visiteurs.

- **Visualisation Petit Train et Repas Possible** : Explorez les options de transport et de restauration disponibles au zoo.

Habitats

- **Fiches Descriptives** : Chaque animal dispose d'une fiche descriptive avec des informations sur son habitat, son alimentation et ses caractéristiques.

Services

- **Plan du Zoo** : Visionnez le plan du zoo pour faciliter votre visite.
- **Services** : Informations sur les services disponibles au zoo, tels que les aires de jeux, les restaurants, et les visites en petit train.

Connexion Employés

- **Formulaire de Connexion avec Droits Spécifiques en Fonction du Poste** :
    - **Employé** : id : employe / mdp : ***
    - **Vétérinaire** : id : veterinaire / mdp : ***
    - **Administrateur** : id : admin / mdp : ***

Note : La connexion et la gestion des accès sont sécurisées grâce aux protocoles de sécurité de mon hébergeur o2switch.

Contact

- **Formulaire de Contact** : Utilisez nos réseaux sociaux ou notre adresse mail pour nous envoyer vos questions ou commentaires.

Footer (Pied de page)

Le pied de page de chaque page du site contient les éléments suivants :

- **Mentions Légales** : En lien direct vers les mentions légales du site, expliquant les responsabilités et droits liés à l'utilisation du site.

- **Conformité RGPD** : Information sur la politique de confidentialité et la gestion des données personnelles en conformité avec la législation RGPD.

Présentation du jeu d’essai pour la vérification de mot de passe

1. Contexte et description de la fonctionnalité :

La fonctionnalité testée concerne la gestion des mots de passe lors de l'authentification des utilisateurs dans le système. Le but était de vérifier si un mot de passe saisi par un utilisateur correspondait au mot de passe enregistré dans la base de données. Pour ce faire, la fonction password_verify() de PHP était utilisée, mais des anomalies ont été détectées lors de l'exécution des tests, car les mots de passe enregistrés n’étaient pas hachés correctement.

2. Données en entrée :

Les données d'entrée pour ce test étaient :

- Identifiant : admin
- **Mot de passe fourni** : Adminpass
- **Mot de passe stocké dans la base de données** : Adminpass mais mal haché

Le test visait à vérifier si la fonction password_verify() pouvait comparer le mot de passe fourni par l'utilisateur avec celui stocké dans la base de données.

3. Données attendues :

Le comportement attendu était que :

- Si le mot de passe fourni par l'utilisateur correspondait au mot de passe enregistré (après vérification via password_verify()), la connexion serait réussie et une session utilisateur serait ouverte.
- Si le mot de passe était incorrect, un message d’erreur indiquant une erreur d’identification devait apparaître.

Dans le cadre d’un bon fonctionnement de password_verify(), le mot de passe en base de données aurait dû être haché, et le test aurait dû vérifier ce hachage contre le mot de passe fourni.

4. Données obtenues :

Lors de l’exécution du test :

- Le mot de passe fourni par l’utilisateur a été comparé avec le mot de passe stocké en clair dans la base de données, mais la fonction password_verify() a renvoyé une erreur.
- L'utilisateur a obtenu une redirection sur la page de connexion indiquant que le mot de passe était incorrect, alors que les deux mots de passe étaient identiques en texte brut.

5. Analyse des écarts :

L'écart observé est dû au fait que la fonction password_verify() attend un mot de passe haché pour effectuer une comparaison. Or, dans ce cas, le mot de passe était enregistré en texte clair dans la base de données, ce qui a causé l’échec de la vérification.

Voici une explication plus détaillée du problème :

- La fonction password_verify() compare le mot de passe fourni par l'utilisateur avec un mot de passe **haché**.
- Dans cette situation, le mot de passe stocké n'était pas haché, mais stocké en texte clair, ce qui a provoqué une incompatibilité avec la méthode de vérification, et donc l'échec de la connexion.

Solution :

- **Hachage des mots de passe lors de l'enregistrement** : Pour résoudre ce problème, tous les mots de passe doivent être hachés lors de leur enregistrement en base de données. La fonction password_hash() de PHP a été utilisée pour cela.

- **Mise à jour de la base de données** : Tous les mots de passe précédemment enregistrés en clair ont été convertis en hachages sécurisés.

Description de la veille de sécurité

1. Veille effectuée :

Durant le développement de cette fonctionnalité, une veille sur les bonnes pratiques de gestion des mots de passe a été effectuée. Cela inclut la surveillance des vulnérabilités de sécurité relatives à la gestion des mots de passe et à l'authentification utilisateur.

2. Vulnérabilités identifiées :

Une des vulnérabilités identifiées concernait le stockage des mots de passe en texte clair ou mal haché dans la base de données. Ce problème constitue une faille majeure car en cas de compromission de la base de données, les mots de passe des utilisateurs peuvent être directement utilisés par des attaquants, sans effort de décryptage. Cela va à l'encontre des recommandations de l'OWASP, qui préconisent de toujours hacher les mots de passe avant leur stockage.

3. Failles corrigées :

Les corrections suivantes ont été apportées :

- **Hachage des mots de passe** : Chaque mot de passe est désormais haché avec password_hash() avant d'être stocké dans la base de données.
- **Utilisation d'algorithmes sécurisés** : Le hachage utilise l'algorithme bcrypt, qui est recommandé pour sa sécurité et sa résistance aux attaques par force brute.

4. Approche proactive :

Pour prévenir d'autres vulnérabilités liées à la gestion des mots de passe, plusieurs mesures proactives ont été mises en place :

- **Mise à jour régulière de sécurité** : Un suivi régulier des mises à jour pour la gestion de la sécurité est effectué.

Conclusion :

La problématique initiale concernant la vérification des mots de passe a mis en lumière une faille importante dans la gestion de la sécurité. Grâce à l’identification rapide de cette vulnérabilité et à l'application de correctifs, la fonctionnalité d’authentification a été sécurisée, et des mesures préventives ont été mises en place pour éviter que des erreurs similaires ne se reproduisent.