Gestion et développement du Projet Web Arcadia

Table des Matières

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.

Développement

Configuration de l'Environnement de Développement

1. **Éditeur de Code**: Ouvrez le projet dans votre éditeur de code préféré.
2. **Serveur Local**: Utilisez une extension de serveur local pour tester les changements en temps réel (par exemple, Live Server pour VS Code).

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">
```

```
<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>
```

```
        <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>
```

```
            </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 :

```
/*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);
}
```

```css
.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);
    }
```

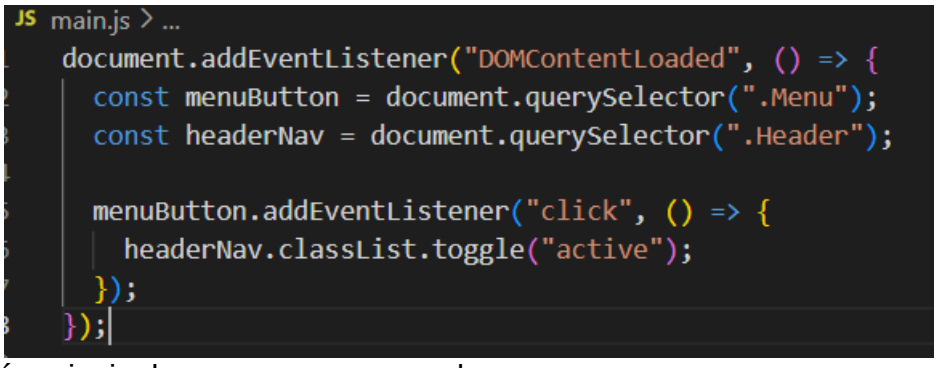
```
    .Header a {
        color: white;
        margin: 0;
        font-size: 1em;
    }
}

@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 :

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.

```
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

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.

```
// 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 :

```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

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');
```

- **Intégration des fichiers complémentaires :**

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

libs > config.php > ...

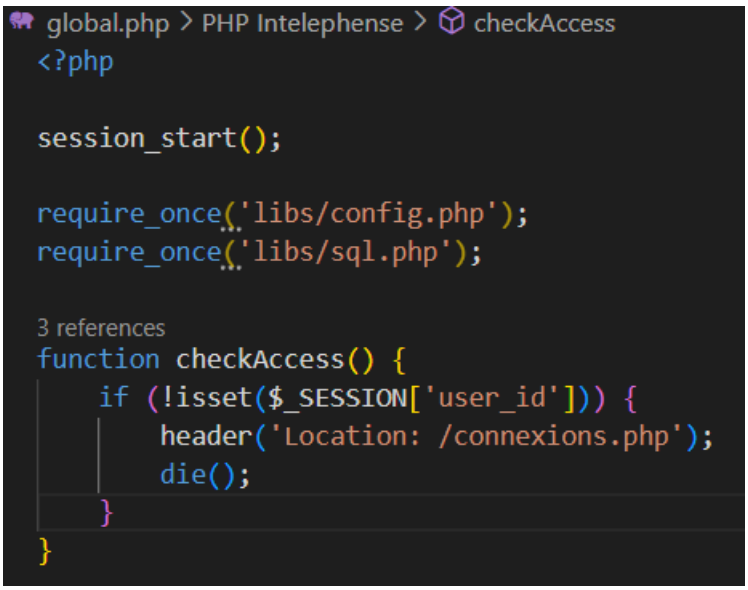
```php
<?php

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 > ...

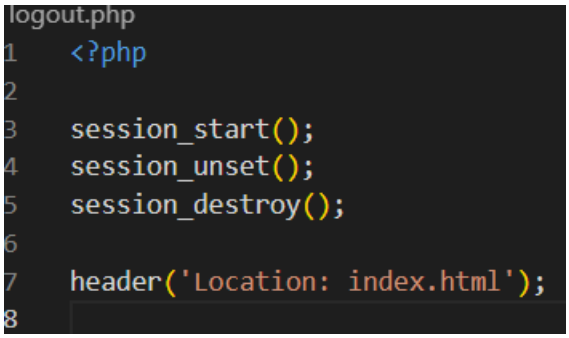
```php
<?php

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.



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`).



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

```
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$10$exemplehash2', 'employe'),
(3, 'veterinaire', '$2y$10$exemplehash3', '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