Bienvenue au Zoo Arcadia

Découvrez un zoo écoresponsable niché au cœur de la légendaire forêt de Brocéliande. Le Zoo Arcadia vous invite à une aventure unique, alliant conservation de la nature et expérience familiale inoubliable.

Notre Mission Verte

Sensibilisation

Nous éduquons nos visiteurs sur l'importance de la conservation et du respect des animaux.

Écoresponsabilité

Notre zoo est énergétiquement indépendant grâce à des solutions durables innovantes.

Bien-être animal

Nous priorisons le confort et la santé de nos pensionnaires dans des habitats naturels.

Un Voyage à Travers les Continents

1

Forêt Tropicale

Découvrez la biodiversité luxuriante des jungles d'Amérique du Sud et d'Asie.

2

Savane Africaine

Observez les majestueux animaux de la savane dans un vaste espace ouvert.

3

Pôle Arctique

Rencontrez les fascinants habitants des régions polaires dans un habitat climatisé.

Une Journée Remplie d'Aventures

Petit Train

Explorez le zoo confortablement à bord de notre train écologique.

Restauration

Savourez des plats locaux et bio dans nos points de restauration.

Aires de Jeux

Les enfants s'amuseront dans nos espaces de jeux thématiques.

Ma Vision pour le Site Web Arcadia

Découvrez comment j'ai conçu un site web captivant et intuitif pour le Zoo Arcadia. De la création graphique à l'intégration de fonctionnalités innovantes, j'ai veillé à offrir une expérience en ligne à la hauteur de l'esprit d'aventure et de préservation de la nature du zoo.

Le Site Web du Zoo Arcadia

Le site web du Zoo Arcadia hebergé sur o2Switch a été conçu avec un design moderne et épuré, mettant en valeur l'expérience de visite au cœur de la nature. Des couleurs vives, des illustrations nature et une navigation intuitive invitent les visiteurs à découvrir les merveilles du parc.

Grâce à une structure claire et des informations pratiques, le site web permet de préparer sa visite en détail et de s'immerger dans l'univers du Zoo Arcadia avant même d'y arriver.

La Charte Graphique

J'ai conçu une charte graphique qui reflète l'esprit naturel et sauvage du Zoo Arcadia. Inspirée par la nature environnante, elle s'articule notamment autour de couleurs vibrantes telles que le vert #445843, un joli dégradé vert plus léger #455e3b et un fond blanc de feuilles. Des illustrations organiques et une typographie épurée complètent cette palette, créant une identité visuelle moderne et accueillante.

Chaque élément - du logo aux éléments de navigation - a été soigneusement conçu pour offrir une expérience de visite cohérente et immersive, de la billetterie en ligne au signalétique sur le site.

Les Outils Techniques Utilisés pour le Site Web Arcadia

Pour construire le site web du Zoo Arcadia, j'ai utilisé un ensemble d'outils et de technologies soigneusement sélectionnés. Ces outils sont des standards de l'industrie, garantissant un développement fiable, efficace et adaptable.

Pour le code, j'ai utilisé VScode, un éditeur de code puissant et polyvalent. Pour la base de données, j'ai utilisé HeidiSQL et PHPmyadmin, deux outils populaires pour gérer et manipuler les données. Pour le serveur, j'ai utilisé XAMPP, une plateforme permettant de tester le site web en environnement local.

Le navigateur Google Chrome a été utilisé pour tester le site web. Pour l'hébergement, j'ai utilisé o2switch, un hébergeur web fiable et performant. Enfin, pour le maquettage et les diagrammes, j'ai utilisé Figma et Lucid, des outils permettant de créer des designs et des schémas visuels professionnels.

Présentation du Code de Mon Site Web

Aujourd'hui, je vais vous présenter le code de mon site web en détaillant les différentes technologies utilisées :

1. **HTML** : Structure de base et organisation des pages.
2. **CSS** : Mise en forme, design et adaptabilité responsive.
3. **JavaScript** : Interactivité et fonctionnalités dynamiques du site.
4. **PHP** : Gestion des requêtes serveur, bases de données et sécurité des sessions.

Cette présentation mettra en lumière comment ces technologies s'intègrent pour créer un site web complet et fonctionnel.

- Ce code représente la section **header** de ma page web, qui contient le logo du site et un menu de navigation permettant aux utilisateurs d'accéder facilement aux différentes sections, comme les services, les habitats, et la connexion des employés.

- Deux fichiers CSS (`main.css` et `accueil.css`) sont inclus pour la mise en page et le style du site.
- Le fichier JavaScript `main.js` est chargé de façon asynchrone grâce à l'attribut `defer`, pour que le script ne bloque pas le chargement du HTML.
- Le contenu principal débute avec une balise `<header>`, contenant un menu de navigation (`<nav>`). Le logo est inséré via une balise `<img>`, et des liens vers d'autres pages sont ajoutés pour la navigation (`<a>`).

```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
                        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
            class="image-panda">
        <div class="text-panda">
```

Style du Header et de la Page

- La règle `html, body` définit un style global pour la page, supprimant les marges par défaut et appliquant une police sans-serif pour assurer une lisibilité optimale.
- Pour l'arrière-plan du site, une image est utilisée (`background-image`) et réglée pour couvrir tout l'écran sans répétition, rester fixe lors du défilement, et centrée pour une apparence équilibrée.
- La section du logo (`#cover`) est stylisée avec une taille fixe, des bordures arrondies, et des ombres pour créer un effet de relief.
- La classe `.navbar` gère l'alignement des éléments du menu en les centrant horizontalement grâce au système flexbox.
- Enfin, la classe `.accueil` applique un style spécifique au texte d'accueil, en augmentant la taille de police et en mettant le texte en gras pour le rendre plus visible.

```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-cont
    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;
```

Styles du Menu et de la Navigation

- L'effet **hover** sur les liens (`a:hover`) change le style de police en italique lors du survol.
- La classe **.Header** gère le menu de navigation : il est centré avec des espaces entre les liens, un fond vert foncé, et des ombres pour un effet de relief.
- Les liens actifs (**.Header a**) simulent un clic.
- La classe **.Menu** est un bouton caché par défaut, destiné à apparaître sur d'autres résolutions plus petites. Il est centré, avec un fond vert, des coins arrondis et un effet de surbrillance grâce aux ombres.

```
    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;
```

Styles Responsifs

- Ce bloc utilise une **media query** pour ajuster l'affichage sur des écrans de moins de 1100 pixels.
- Le bouton **.Menu** devient visible grâce à `display: flex` et est centré.
- La barre de navigation **.Header** est cachée sur ces petits écrans (`display: none`), mais elle devient un menu fixe qui s'affiche en colonne avec un fond vert et des liens centrés lorsque le bouton menu est activé.

```
    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;
```

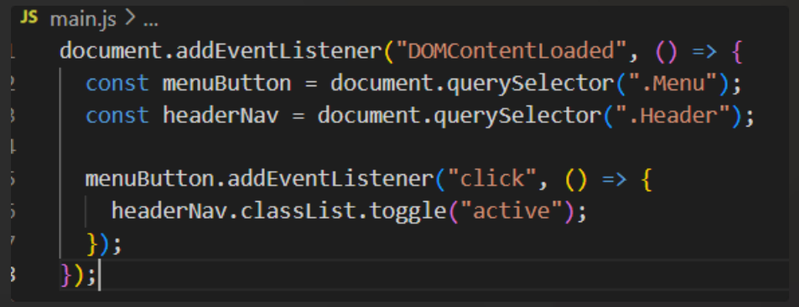
Styles pour Mobile et Footer

- La **media query** pour les écrans de moins de 480 pixels ajuste le menu **.Header** quand il est activé : il devient un menu en ligne, centré et adapté à la largeur de l'écran.
- La classe **.Mentions** gère le style du pied de page (footer), avec un texte en grand format centré, espacé avec `gap`, et un fond vert avec des ombres pour un effet de profondeur. Les éléments sont bien alignés et organisés en utilisant **flexbox**.

```
    .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;
    }
}

/*footer*/
.Mentions {
    display: flex;
    font-size: x-large;
    justify-content: center;
    gap: 40px;
    padding: 0 20px 0 20px;
    width: auto;
    height: 170px;
    border-radius: 10px;
    background-color: #445843;
    align-items: center;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
    box-shadow: 4px 4px 6px rgba(0, 0, 0, 0.4);
}
```



Interactivité du Menu

- Ce script attend que la page soit entièrement chargée (`DOMContentLoaded`).
- Il récupère le bouton menu (`.Menu`) et la barre de navigation (`.Header`).
- Lorsque l'utilisateur clique sur le bouton menu, la classe `active` est ajoutée ou retirée (`toggle`), ce qui déclenche l'affichage ou le masquage du menu de navigation sur les petits écrans.

Interactions Avancées

- **Chargement de la Page** : Le script attend que la page soit complètement chargée avant de s'exécuter.
- **Menu** : Lorsque le bouton menu (`.Menu`) est cliqué, la classe `active` est ajoutée ou retirée de la barre de navigation (`.Header`), affichant ou cachant le menu.
- **Cartes** : Chaque carte (`.card`) réagit au toucher. Lorsqu'une carte est touchée, elle reçoit la classe `active`, et toutes les autres cartes la perdent.
- **Toucher en Dehors** : Si l'utilisateur touche en dehors d'une carte, toutes les cartes perdent la classe `active`.

ervices.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
      card.classList.add("active"); // Ajouter la cla
    });
  });

  document.addEventListener("touchstart", function (e
    if (!event.target.closest(".card")) {
      cards.forEach((card) => card.classList.remove('
    }
  });
});
```

Gestion des Interactions et AJAX

- **Chargement de la Page** : Le script s'exécute une fois la page entièrement chargée.
- **Menu** : Le bouton menu (`.Menu`) contrôle l'affichage du menu de navigation (`.Header`) en ajoutant ou retirant la classe `active` lorsqu'il est cliqué.
- **Fonction** `toggleListe` : Cette fonction gère l'affichage des listes associées aux boutons :
    - Lorsqu'un bouton est cliqué, il empêche le comportement par défaut, vérifie si la liste est visible, et si ce n'est pas le cas, soumet le formulaire via AJAX.
    - Utilise `fetch` pour envoyer les données du formulaire en arrière-plan, puis affiche un message de succès ou d'erreur selon la réponse du serveur.
    - Enfin, elle bascule la visibilité de la liste associée.
- **Initialisation** : Les boutons (`.foret`, `.savane`, `.foret-montagneuse`, `.antarctique`) sont reliés à leurs listes correspondantes pour activer cette fonctionnalité

```
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");
});
```

libs > config.php > ...

```php
<?php

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

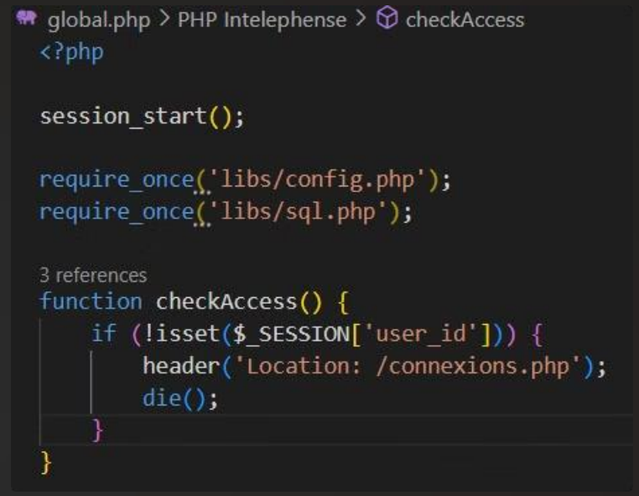
Configuration de la Base de Données

- **Définition des Constantes :**
  - `DB_HOST` : Spécifie l'hôte de la base de données (ici `localhost`).
  - `DB_NAME` : Nom de la base de données à utiliser (`arcadia1` en local).
  - `DB_USER` : Nom d'utilisateur pour se connecter à la base de données (`root` en local).
  - `DB_PASSWORD` : Mot de passe associé à l'utilisateur (`root` en local).

Il faut ensuite modifier les paramètres en fonction de l'hébergeur pour la mise en production.

Gestion des Sessions et Vérification d'Accès

- **Démarrage de la Session** : `session_start()` initialise ou reprend une session PHP, permettant de gérer les variables de session.
- **Inclusion des Fichiers** : Les fichiers `config.php` et `sql.php` sont inclus pour accéder aux configurations et aux fonctions SQL nécessaires.
- **Fonction** `checkAccess` : Vérifie si un utilisateur est connecté :
    - Si la variable de session `user_id` n'est pas définie, l'utilisateur est redirigé vers la page de connexion (`/connexions.php`) et le script s'arrête (`die()`).



Fonction `initConnexion`

- **Création de la Connexion** : Initialise une connexion à la base de données en utilisant les constantes définies (`DB_HOST`, `DB_USER`, `DB_PASSWORD`, `DB_NAME`).
- **Configuration du Charset** : Définit le jeu de caractères de la connexion à `utf8mb4` pour garantir la prise en charge des caractères spéciaux.
- **Vérification des Erreurs** : Vérifie si la connexion échoue. Si c'est le cas, un message d'erreur est affiché et le script s'arrête (`die()`).
- **Retour de la Connexion** : Retourne l'objet de connexion `$conn` pour une utilisation ultérieure dans le script.

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;
}

```

Explication :

1. `session_start();`
    - **Démarre** ou **reprend** une session existante.
2. `session_unset();`
    - **Efface** toutes les **variables de session**.
3. `session_destroy();`
    - **Détruit** la **session** complètement.
4. `header('Location: index.html');`
    - **Redirige** l'utilisateur vers `index.html`.
5. `exit();`
    - **Arrête** l'exécution du script après la redirection.

logout.php

```php
<?php

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

header('Location: index.html');
exit();
```

Traitement de la Connexion

- **Inclusion du Fichier de Configuration** : Le fichier `global.php` est inclus pour charger les configurations nécessaires.
- **Création et Vérification de la Connexion** : Une connexion à la base de données est établie avec `mysqli`. Si la connexion échoue, un message d'erreur est affiché et le script s'arrête (`die()`).
- **Récupération des Données du Formulaire** : Les données d'identifiant et de mot de passe sont récupérées et nettoyées avec `trim()`.
- **Préparation de la Requête SQL** : Une requête SQL sécurisée (`SELECT`) est préparée pour récupérer les informations utilisateur en fonction de l'identifiant.
- **Exécution et Vérification** : La requête est exécutée. Si aucun utilisateur n'est trouvé, une erreur est stockée dans la session et l'utilisateur est redirigé vers la page de connexion.
- **Vérification du Mot de Passe** : Le mot de passe soumis est vérifié par rapport au mot de passe haché dans la base de données. Si le mot de passe est incorrect, une erreur est stockée dans la session et l'utilisateur est redirigé.

nexion.php > ...

```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 userr
$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 incorre
    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 incorre
    header('Location: /zooarcadia/connexions.php');
```

Gestion de l'Authentification Réussie

- **Sécurisation de la Session** : `session_regenerate_id(true)` crée un nouvel ID de session pour protéger contre les attaques de fixation de session.
- **Stockage des Informations de Session** : Les identifiants de l'utilisateur et son rôle sont stockés dans la session (`$_SESSION`).
- **Redirection en Fonction du Rôle** :
  - Si l'utilisateur est un **vétérinaire**, il est redirigé vers `connectveto.php`.
  - Si l'utilisateur est un **administrateur**, il est redirigé vers `admin.php`.
  - Si l'utilisateur est un **employé**, il est redirigé vers `employe.php`.
  - Si le rôle est inconnu, il est redirigé vers la page de connexion (`connexions.php`).
- **Fin du Script** : `exit()` est utilisé pour arrêter l'exécution du script après la redirection.

```
38      exit(); // Ajout de exit() pour stopper l'exécution
39  }
40
41  // Authentification réussie
42  session_regenerate_id(true); // Sécuriser la session
43  $_SESSION['user_id'] = $user['user_id'];
44  $_SESSION['role'] = $user['role'];
45
46  // Redirection basée sur le rôle
47  if ($user['role'] === 'veterinaire') {
48      header('Location: /zooarcadia/connectveto.php');
49  } elseif ($user['role'] === 'admin') {
50      header('Location: /zooarcadia/admin.php');
51  } elseif ($user['role'] === 'employe') {
52      header('Location: /zooarcadia/employe.php');
53  } else {
54      header('Location: /zooarcadia/connexions.php');
55  }
56  exit(); // Ajout de exit() pour stopper l'exécution
57
```

Traitement du Formulaire d'Ajout de Données

- **Inclusion et Vérification** : Le fichier `global.php` est inclus et la fonction `checkAccess()` vérifie que l'utilisateur est authentifié.
- **Initialisation de la Connexion** : La connexion à la base de données est établie via `initConnexion()`.
- **Traitement du Formulaire** :
    - Si la méthode de requête est POST et que tous les champs requis sont présents (`animal_id`, `quantity`, `checkup_date`, `health`), les données sont nettoyées et préparées pour l'insertion.
    - **Requête SQL** : Prépare et exécute une requête d'insertion sécurisée dans la table `animal_data`.
    - En cas d'échec de la préparation ou de l'exécution de la requête, un message d'erreur est affiché.
- **Réussite** : Si l'insertion réussit, une variable de session `success` est définie pour indiquer le succès.
- **Redirection** : L'utilisateur est redirigé vers la même page pour éviter la soumission multiple du formulaire et pour afficher les données mises à jour.

```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();
```

Traitement des Formulaires et Requêtes SQL

- **Réinitialisation des Clics** :
  - Si le formulaire de réinitialisation est soumis (`$_POST['reset']`), une requête SQL est préparée pour réinitialiser les clics dans la table `habitat_clicks`.
  - En cas d'erreur lors de la préparation ou de l'exécution de la requête, un message d'erreur est affiché.
  - Après une réinitialisation réussie, l'utilisateur est redirigé vers la même page pour mettre à jour les données affichées.
- **Sélection des Clics des Habitats** :
  - Une requête SQL est préparée pour sélectionner tous les enregistrements de la table `habitat_clicks`.
  - Les résultats sont récupérés et stockés dans un tableau associatif (`$clicksData`).
- **Sélection des Données de Soins des Animaux** :
  - Une requête SQL est préparée pour sélectionner les données de soins des animaux en joignant la table `animal_data` avec `users` pour obtenir les détails des utilisateurs. Les résultats sont triés par date de consultation (`checkup_date`) dans l'ordre décroissant.

```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();
```

Exécution de la Requête SQL

- **Préparation de la Requête** : La requête SQL (`$sql`) est préparée pour éviter les injections SQL. Si la préparation échoue, un message d'erreur est affiché.
- **Exécution de la Requête** : La requête est exécutée avec `$stmt->execute()`.
- **Récupération des Résultats** : Les résultats de la requête sont récupérés avec `$stmt->get_result()` et stockés dans `$result`.
- **Fermeture des Ressources** : La déclaration (`$stmt`) et la connexion (`$conn`) sont fermées pour libérer les ressources.

Intégration Firebase et Suivi des Clics

- **Importation des Modules Firebase** : Le code importe les modules nécessaires pour utiliser Firebase et Firestore à partir des URLs spécifiées.
- **Configuration Firebase** : `firebaseConfig` contient les informations nécessaires pour connecter l'application à Firebase.
- **Initialisation de Firebase** : `initializeApp(firebaseConfig)` initialise l'application Firebase, et `getFirestore(app)` obtient une instance de Firestore pour interagir avec la base de données.
- **Enregistrement des Clics** :
  - **Fonction** `logClick` : Enregistre les clics dans la collection `clicks` de Firestore avec le nom de la plateforme (`Instagram` ou `Facebook`) et l'horodatage.
  - **Gestion des Événements** : Lorsqu'un lien dans la section `.formcontact` est cliqué, la fonction `logClick` est appelée avec la plateforme appropriée.
- **Gestion des Erreurs** : Les erreurs lors de l'ajout d'un document sont capturées et affichées dans la console.

```
<script type="module">
    import { initializeApp } from "https://www.gstatic.com/firebasejs/9.9.4/fire
    import { getFirestore, collection, addDoc } from "https://www.gstatic.com/fi

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

    // Initialisation 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')
                logClick(platform);
            });
        });
```

base de données

je vais maintenant vous expliquer comment la base de données de mon site web est structurée et gérée :

- **Modèle de Données** : Organisation des tables et des relations pour structurer efficacement les informations.
- **Requêtes SQL et utilisation de l'opération CRUD** : Utilisation des requêtes dont **Create** pour ajouter de nouvelles données.

```sql
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;
```

```sql
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;
```

```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;
```

```sql
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;
```

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐	1	**id** 🔑	int(11)			Non	*Aucun(e)*		AUTO_INCREMENT	Modifier Supprimer Plus
☐	2	**animal**	varchar(255)	utf8mb4_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	3	**quantity**	int(11)			Non	*Aucun(e)*			Modifier Supprimer Plus
☐	4	**checkup_date**	date			Non	*Aucun(e)*			Modifier Supprimer Plus
☐	5	**health**	text	utf8mb4_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	6	**user_id**	int(11)			Non	*Aucun(e)*			Modifier Supprimer Plus

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
☐	1	**id** 🔑	int(11)			Non	*Aucun(e)*		AUTO_INCREMENT
☐	2	**pseudo**	varchar(100)	utf8mb4_general_ci		Non	*Aucun(e)*		
☐	3	**texte**	text	utf8mb4_general_ci		Non	*Aucun(e)*		
☐	4	**date**	timestamp			Non	current_timestamp()		

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
☐	1	**id** 🔑	int(11)			Non	*Aucun(e)*		AUTO_INCREMENT
☐	2	**habitat** 🔗	varchar(255)	utf8mb4_general_ci		Non	*Aucun(e)*		
☐	3	**clicks**	int(11)			Oui	0		

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
☐	1	**user_id** 🔑	int(11)			Non	*Aucun(e)*		AUTO_INCREMENT
☐	2	**username** 🔗	varchar(50)	utf8mb4_general_ci		Non	*Aucun(e)*		
☐	3	**password**	varchar(255)	utf8mb4_general_ci		Non	*Aucun(e)*		
☐	4	**role**	enum('admin', 'employe', 'veterinaire')	utf8mb4_general_ci		Non	*Aucun(e)*		

Structure de mes tables

Sécurité du Site Web Arcadia

Sécurité et Authentification dans le Code PHP

- **Requêtes Préparées** : Utilisation de requêtes préparées pour interagir avec la base de données, empêchant les injections SQL.
- **Hashage des Mots de Passe** : Vérification sécurisée des mots de passe avec `password_verify()` contre les mots de passe hachés en base de données.
- **Gestion des Sessions** : Sécurisation des sessions par la régénération de l'ID après connexion (`session_regenerate_id(true)`), avec stockage sécurisé des informations d'utilisateur.

Sécurité du Site Web Arcadia

Fonctionnalités Clés :

1. **Démarrage de la Session** : `session_start();` au début du script.
2. **Nettoyage des Données d'Entrée** : Utilisation de `trim()` pour supprimer les espaces superflus.
3. **Préparation des Requêtes SQL** : Préparation et liaison des paramètres pour éviter les injections SQL.
4. **Vérification de l'Utilisateur et du Mot de Passe** : Validation de l'existence de l'utilisateur et utilisation de `password_verify()` pour comparer les mots de passe.
5. **Régénération de l'ID de Session** : Renouvellement de l'ID après une connexion réussie pour empêcher les détournements de session.
6. **Redirection Basée sur le Rôle** : Redirection des utilisateurs en fonction de leur rôle (admin, vétérinaire, employé).
7. **Utilisation de `exit()`** : Arrêt immédiat du script après redirection pour éviter tout traitement postérieur non désiré.

Ces mesures garantissent la sécurité des données et des interactions utilisateurs tout en protégeant le système contre les attaques.

Maquettage du Site Web Arcadia

Figma pour le Design

J'ai utilisé l'outil de design Figma pour créer des maquettes haute-fidélité du site web Arcadia. Cela m'a permis d'explorer différentes options de mise en page, de typographie et de couleurs afin de définir l'identité visuelle du zoo.

Wireframes avec Lucid

Pour structurer l'expérience utilisateur, j'ai réalisé des wireframes avec l'outil Lucid. Cela m'a aidé à définir l'architecture de l'information et le cheminement des visiteurs sur le site, de la billetterie en ligne aux pages de contenus.

Conclusion: Un Site Web efficace pour Arcadia

Le site web du zoo Arcadia permet de présenter le zoo aux visiteurs, de promouvoir ses missions et de fidéliser le public. Il est moderne, accessible et sécurisé pour répondre aux besoins des visiteurs et des employés.

Grâce aux technologies utilisées, le site web est performant, convivial et attractif. Des maquettes professionnelles et des outils de design modernes garantissent une expérience utilisateur optimale.

Le site web sera un véritable reflet de la mission et des valeurs du zoo Arcadia. Il encouragera les visiteurs à découvrir le monde fascinant de la faune et à s'engager pour sa protection.

Merci pour votre attention !

Je vous remercie d'avoir pris le temps de découvrir le site web du Zoo Arcadia. J'espère avoir pu éveiller votre intérêt et votre curiosité pour mon site web.