

NOM	JEGO
Prénom	Guillaume
Date de naissance	02/11/1990

Copie à rendre

TP – Développeur Web et Web Mobile

Documents à compléter et à rendre

Lien du git : <https://github.com/GuillaumeJego/arcadia-CCF-JEGO.git>

Lien de l'outil de gestion de projet : (dans le repo git, FIGMA)

Lien du déploiement : Local MAMP

Login et mot de passe administrateur : Inséré en dur pour l'examen dans le code php. Vue que j'utilise une BDD local et que j'ai limité les manipulations des examinateurs à créer des variables d'environnements dans les variables d'environnement du compte utilisateur de l'examineur. Ils sont indiqués dans le fichier readme

```
Utilisateur user1  
Mot de passe : zUWr/KrCIKsGxpy9
```

## SANS CES ELEMENTS, VOTRE COPIE SERA REJETEE

### Partie 1 : Analyse des besoins

1. Effectuez un résumé du projet en français d'une longueur d'environ 20 lignes soit 200 à 250 mots
2. Exprimez le cahier des charges, l'expression du besoin ou les spécifications fonctionnelles du projet

### Partie 2 : Spécifications technique

1. Spécifiez les technologies que vous avez utilisé en justifiant les conditions d'utilisation et pourquoi le choix de ses éléments
2. Comment avez-vous mis en place votre environnement de travail ? Justifiez vos choix. (README.md)
3. Énumérez les mécanismes de sécurité que vous avez mis en place, aussi bien sur vos formulaires que sur les composants front-end ainsi que back-end.

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

## Partie 3 : Recherche

1. Décrivez une situation de travail ayant nécessité une recherche durant le projet à partir de site anglophone. N'oubliez pas de citer la source
2. Mentionnez l'extrait du site anglophone qui vous a aidé dans la question précédente en effectuant une traduction en français.

## Partie 4 : Informations complémentaire

1. Autres ressources
2. Informations complémentaires

## Table des matières

Partie 1 : Analyse des besoins .....	5
1-1°) Effectuez un résumé du projet en français d'une longueur d'environ 20 lignes soit 200 à 250 mots.....	5
1-2°) Exprimez le cahier des charges, l'expression du besoin ou les spécifications fonctionnelles du projet .....	6
I°) Cahier des Charges.....	6
II°) Expression du Besoin .....	7
III°) Spécifications Fonctionnelles.....	7
Partie 2 : Spécifications techniques.....	9
2-1°) Spécifiez les technologies que vous avez utilisé en justifiant les conditions d'utilisation et pourquoi le choix de ses éléments.....	9
Front-End : .....	9
Back-End : .....	9
2-2°) Comment avez-vous mis en place votre environnement de travail ? Justifiez vos choix. (README.md).....	11
2-3°) Énumérez les mécanismes de sécurité que vous avez mis en place, aussi bien sur vos formulaires que sur les composants front-end ainsi que back-end.....	12
2-4°) Décrivez une veille technologique que vous avez effectuée, sur les vulnérabilités de sécurité. ....	16
Partie 3°) recherches .....	19
3-1°) Décrivez une situation de travail ayant nécessité une recherche durant le projet à partir de site anglophone. N'oubliez pas de citer la source.....	19
3-2°) Mentionnez l'extrait du site anglophone qui vous a aidé dans la question précédente en effectuant une traduction en français.....	20
Partie 4 : Informations complémentaire .....	21
4-1°) Autres ressources .....	21
4-2°) Informations complémentaires .....	21

## Partie 1 : Analyse des besoins

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

Bonjour,

Ce projet a pour objectif de développer une application web pour un zoo, visant à répondre à deux besoins principaux.

**Premièrement**, l'application est destinée à améliorer la gestion interne du zoo. Le client souhaite disposer d'un meilleur suivi des animaux et d'une traçabilité complète des actions réalisées par le personnel sur l'ensemble des éléments du zoo, incluant les structures et les animaux. Cette fonctionnalité permettra de centraliser les informations et de faciliter la gestion quotidienne, garantissant ainsi un suivi précis et efficace.

**Deuxièmement**, l'application vise à améliorer la communication avec les clients du zoo. Cette communication externe a pour but d'attirer de nouveaux visiteurs en présentant les services offerts par le zoo, ainsi que ses horaires d'ouverture. Une fois sur place, les visiteurs pourront utiliser l'application pour suivre la routine des animaux, obtenir des informations sur leurs origines et en apprendre davantage sur leur traitement. Le zoo accorde une grande importance au bien-être des animaux et veut mettre en avant ses efforts pour réduire l'impact écologique de ses activités.

**En résumé**, ce projet d'application web vise à optimiser la gestion interne du zoo tout en améliorant la communication externe pour attirer et informer les visiteurs, tout en mettant en avant les valeurs éthiques et écologiques du zoo. Ces deux aspects combinés permettront d'offrir une expérience enrichissante et transparente pour les clients, tout en garantissant une gestion efficace et respectueuse de l'environnement.

## 1-2°) Exprimez le cahier des charges, l'expression du besoin ou les spécifications fonctionnelles du projet

Pour rédiger l'expression du cahier des charges, l'expression du besoin et les spécifications fonctionnelles du projet pour le zoo Arcadia, je vais structurer les informations disponibles dans le document en trois sections distinctes.

### 1°) Cahier des Charges

#### *1. Présentation du Projet*

Le projet vise à développer une application web pour le zoo Arcadia, situé près de la forêt de Brocéliande, en Bretagne. L'objectif est de fournir une solution numérique qui améliore la gestion interne du zoo et la communication avec les visiteurs.

#### *2. Objectifs*

- Gestion interne : Centraliser et faciliter la gestion des animaux et des actions du personnel.
- Communication externe : Attirer de nouveaux visiteurs et offrir une expérience enrichissante aux visiteurs présents.

#### *3. Technologies Utilisée*

- Front-end : HTML5, CSS (Bootstrap), Angular.
- Back-end : PHP avec utilisation de PDO.
- Base de données relationnelle : MySQL, MariaDB.
- Déploiement : local pour l'examen (J'aurais envisagé sur cloud Amazon pour un réel déploiement avec gitZac).

#### *4. Livrables*

- Code source sur GitHub. (Lien page 2 de ce présent document)
- Application à déployer en local. (Application déployée sur cloud Amazon pour un cas réel avec gitZac)
- Documentation de gestion de projet et technique.
- Manuel d'utilisation.
- Charte graphique et maquettes. (dans le repo git)

## II°) Expression du Besoin

### 1. Contexte

Le zoo Arcadia souhaite moderniser ses outils de gestion et de communication pour améliorer l'expérience des visiteurs et la gestion interne du zoo. José, le directeur, souhaite une application web qui reflète les valeurs **écologiques** du zoo.

### 2. Besoins Identifiés

- Interface utilisateur : Doit être simple et intuitive.
- Gestion des animaux : Suivi des animaux, alimentation, état de santé.
- Communication visiteurs : Présentation des services, horaires, avis des visiteurs.
- Administration : Gestion des utilisateurs, services, horaires, habitats, et animaux.

### 3. Objectifs Spécifiques

- Améliorer la gestion des animaux : Traçabilité et suivi détaillé.
- Renforcer la communication : Attirer de nouveaux visiteurs et informer les visiteurs sur place.
- Valoriser l'image **écologique** : Interface et thème **écologiques**.

## III°) Spécifications Fonctionnelles

### 1. Utilisateur Visiteur

- Page d'accueil : Présentation du zoo, images, habitats, services, avis.
- Menu de navigation : Accès aux services, habitats, connexion, contact.
- Vue des services : Liste des services proposés avec descriptions.
- Vue des habitats : Liste des habitats et des animaux associés, détails sur les animaux et avis des vétérinaires.
- Soumission d'avis : Formulaire pour laisser des commentaires.

### 2. Utilisateur Administrateur

- Gestion des comptes : Création et gestion des comptes employé et vétérinaire.
- Gestion du contenu : Modification des services, horaires, habitats, et animaux.
- Dashboard : Statistiques sur la consultation des animaux.

### 3. Utilisateur Employé

- Validation des avis : Valider ou invalider les avis des visiteurs.
- Gestion de l'alimentation : Enregistrement de la nourriture donnée aux animaux.

### 4. Utilisateur Vétérinaire

- Comptes rendus : Saisie des états de santé et des soins des animaux.
- Commentaires sur les habitats : Évaluations et recommandations sur les habitats.

## 5. Connexion et Sécurité

- Authentification : **Accès sécurisé** pour les administrateurs, vétérinaires, et employés.
  - (Pas encore déployé)
- Gestion des accès : Différents niveaux d'accès selon le rôle de l'utilisateur.
  - (Pas encore déployé)



## Partie 2 : Spécifications techniques

2-1°) Spécifiez les technologies que vous avez utilisé en justifiant les conditions d'utilisation et pourquoi le choix de ses éléments

Front-End :

### Le Framework :

- J'ai utilisé le Framework Angular sur sa version 17.0.1
- J'ai utilisé la version 20.9.0 de node.js
- J'ai utilisé la version 10.1.0 de npm

```
PS C:\Users\jegog> ng version

Angular CLI
Angular CLI: 17.0.1
Node: 20.9.0
Package Manager: npm 10.1.0
OS: win32 x64

Angular:
...

Package      Version
-----
@angular-devkit/architect 0.1700.1 (cli-only)
@angular-devkit/core      17.0.1 (cli-only)
@angular-devkit/schematics 17.0.1 (cli-only)
@schematics/angular       17.0.1 (cli-only)

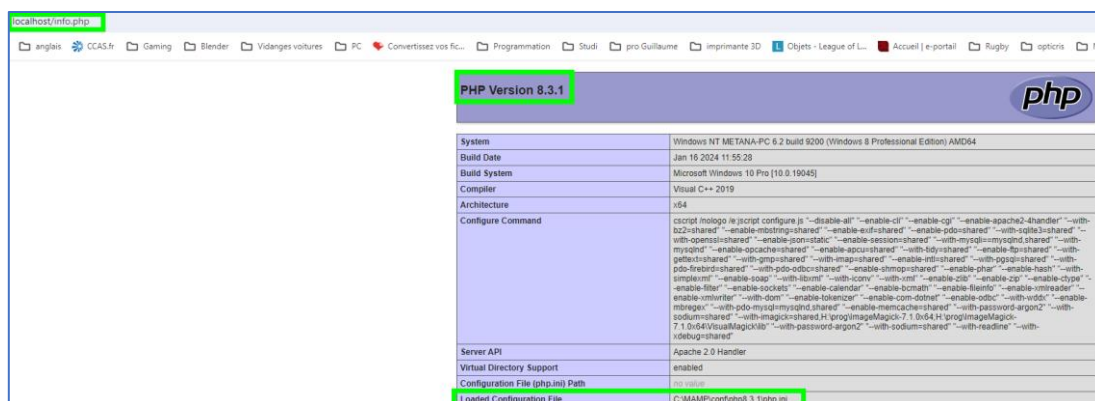
PS C:\Users\jegog> node -v
v20.9.0
PS C:\Users\jegog> npm -v
10.1.0
PS C:\Users\jegog> |
```

### Technologies utilisées dans ce Framework :

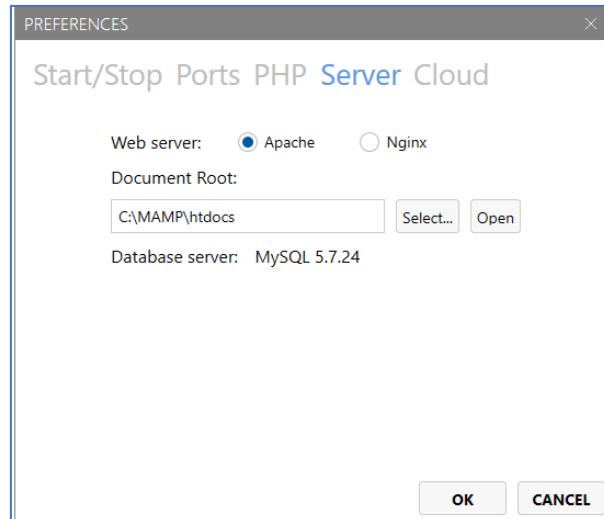
- HTML
- SCSS
- TypeScript

Back-End :

- Php 8.3.1



- Base de données en local via MAMP avec phpMyAdmin
- Serveur base de données MySQL server 5.7.24



- Serveur web : apache 2.4.33

Configuration	
apache2handler	
Apache Version	Apache/2.4.33 (Win64) OpenSSL/1.0.2u mod_fcgid/2.3.9 PHP/8.3.1
Apache API Version	20120211

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

```

1 # Arcadia
2
3 # 1° Installer MAMP (si vous ne l'avez pas déjà)
4 https://www.mamp.info/en/downloads/
5 # vérifiez dans les préférences de MAMP que le serveur est bien coché sur Apache.
6 # Vérifiez que les ports sont les suivants :
7 # Apache Port: 8888
8 # Nginx Port : 7888
9 # MySQL Port : 8889
10 # Démarrez le serveur MAMP : (Start)
11
12 # 2° Ouvrez un terminal et vous placez à la racine de htdocs
13 # Sous Windows
14 cd C:\WAMP\htdocs
15
16 # 3° cloner le repo sur votre disque local
17 # Commande cli :
18 git clone https://github.com/GuillaumeJego/arcadia-CCF-JEGO.git
19 # (Bien vous assurer de copier le dossier et le coller C:\WAMP\htdocs\phpMyAdminDossier à la racine de htdocs)
20
21 # 4° ouvrir le dossier tout juste cloné
22 # Dans le terminal, tapez la ligne d ecommande suivante :
23 cd .\arcadia-CCF-JEGO\
24
25 # 5° Tapez la commande suivante pour déplacer le dossier à la racine de htdoc
26 Move-Item -Path "C:\WAMP\htdocs\arcadia-CCF-JEGO\phpMyAdminDossier" -Destination "C:\WAMP\htdocs\phpMyAdminDossier"
27
28 # Vous devriez toujours vous situer ici :
29 # C:\WAMP\htdocs\arcadia-CCF-JEGO\
30
31 # 6° Maintenant taper la ligne de commande dans votre terminal pour installer les dépendances d'Angular
32 npm i
33
34 # 7° Taper maintenant ng serve pour lancer le serveur local
35 ng serve
36
37 # 8° Le terminal vous affichera le lien pour accéder à l'application en simulation web.
38 # c'est généralement :
39 Local: http://localhost:4200/
40
41 # base de donnée : lien phpMyAdmin ;
42 http://localhost:8888/phpMyAdminDossier/
43
44 # Nom utilisateur :
45 user1
46
47 # mot de passe :
48 zUmr/KrClKSGxpyj
49
50 Les éléments qui n'ont permis d'alimenter l'application se situe dans
51 htdocs\arcadia-CCF-JEGO\src\assets\Images
52
53 This project was generated with [Angular CLI](https://github.com/angular/angular-cli) version 17.0.1.
54
55 ## Development server
56
57 Run 'ng serve' for a dev server. Navigate to 'http://localhost:4200/'. The application will automatically reload if
58
59 ## Code scaffolding
60
61 Run 'ng generate component component-name' to generate a new component. You can also use 'ng generate directive|pipe
62
63
64

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Server bundles	Names	Raw size
Initial chunk files		
chunk-DFQMBL62.mjs	-	1.71 MB
polyfills.server.mjs	polyfills.server	557.44 kB
main.server.mjs	main.server	468.81 kB
chunk-VPSODEBN.mjs	-	2.51 kB
render-utils.server.mjs	render-utils.server	423 bytes

Lazy chunk files	Names	Raw size
chunk-OTT6LQSK.mjs	xhr2	39.10 kB

Application bundle generation complete. [3.834 seconds]

Watch mode enabled. Watching for file changes...

→ Local: http://localhost:4200/

→ press h + enter to show help

2-3°) Énumérez les mécanismes de sécurité que vous avez mis en place, aussi bien sur vos formulaires que sur les composants front-end ainsi que back-end.

Voici une liste des mécanismes de sécurité que j'ai mis en place dans mes formulaires front-end Angular et dans mon back-end PHP, ainsi que les améliorations supplémentaires futures pour renforcer la sécurité.

## Sécurité mise en place dans le Front-end

### 1. Validation des Formulaires :

- Utilisation de la validation des formulaires Angular pour s'assurer que les champs obligatoires sont remplis et que les données saisies sont conformes aux attentes (ex. : utilisation de `required` pour les champs obligatoires).

```

1  <!-- Permet d'ajouter un petit formulaire pour ajouter les cartes des animaux -->
2  <form (ngSubmit)="onSubmit()">
3    <label for="prenom">Prenom:</label>
4    <input
5      type="text"
6      id="prenom"
7      [(ngModel)]="animal.prenom"
8      name="prenom"
9      required
10  >
  
```

### 2. Sanitisation des Entrées :

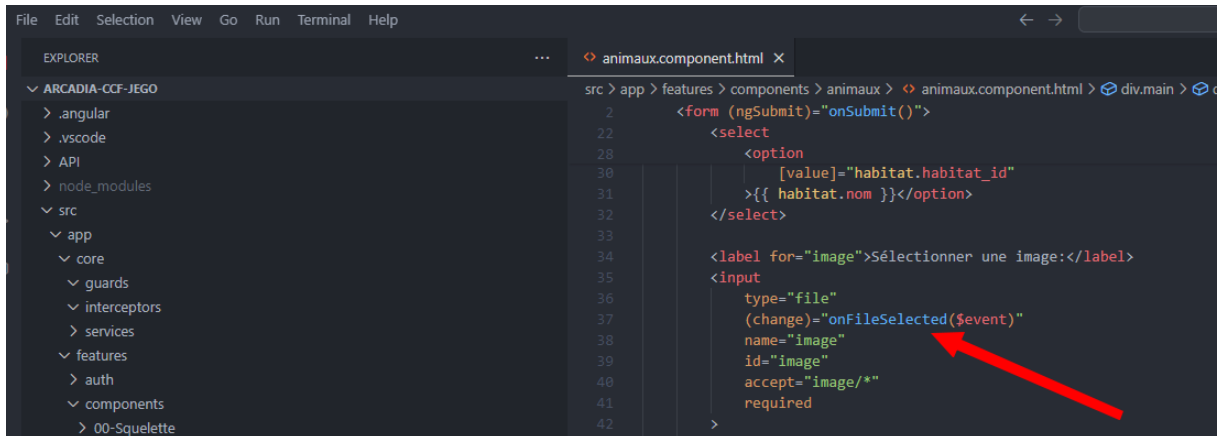
- Angular fournit automatiquement une protection contre les attaques XSS (Cross-Site Scripting) en sanitising les bindings et les templates avec l'utilisation des bindings `{{ Variable }}` pour insérer du contenu dynamique de manière sécurisée.

```

50 <app-habitats></app-habitats>
51
52 <div class="main">
53   <div class="container">
54     <div
55       *ngFor="let animal of animaux"
56       class="tableauAnimaux"
57     >
58       <div
59         *ngIf="animal.habitat_id === selectedHabitatId"
60         class="cardAnimal"
61       >
62         <h1>{{ animal.prenom }}</h1>
63         <a class="image-text">
64           <div class="image-container">
  
```

## 3. Gestion des Fichiers :

- Utilisation de la méthode `onFileSelected` pour gérer les fichiers de manière sécurisée, en redimensionnant les images avant de les envoyer au serveur pour éviter les attaques par déni de service via des fichiers volumineux.



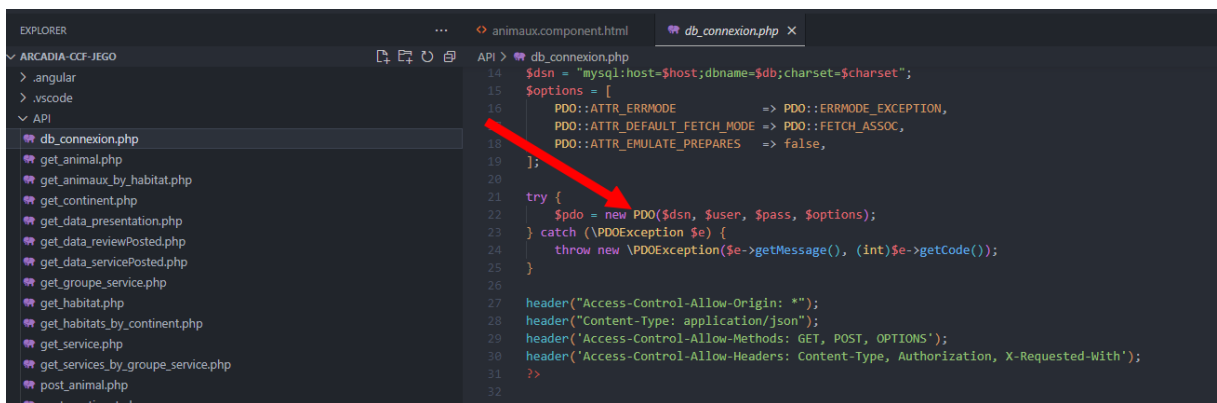
```

2      <form (ngSubmit)="onSubmit()"
22      <select
28      <option
30          [value]="habitat.habitat_id"
31          >{{ habitat.nom }}</option>
32      </select>
33
34      <label for="image">Sélectionner une image:</label>
35      <input
36          type="file"
37          (change)="onFileSelected($event)"
38          name="image"
39          id="image"
40          accept="image/*"
41          required
42  
```

## Sécurité dans le Back-end PHP

### 1. Connexion à la Base de Données avec PDO :

- Utilisation de PDO (PHP Data Objects) pour se connecter à la base de données, ce qui permet d'utiliser des requêtes préparées et de se protéger contre les attaques par injection SQL.



```

14  $dsn = "mysql:host=$host;dbname=$db;charset=$charset";
15  $options = [
16      PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
17      PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
18      PDO::ATTR_EMULATE_PREPARES   => false,
19  ];
20
21  try {
22      $pdo = new PDO($dsn, $user, $pass, $options);
23  } catch (\PDOException $e) {
24      throw new \PDOException($e->getMessage(), (int)$e->getCode());
25  }
26
27  header("Access-Control-Allow-Origin: *");
28  header("Content-Type: application/json");
29  header("Access-Control-Allow-Methods: GET, POST, OPTIONS");
30  header("Access-Control-Allow-Headers: Content-Type, Authorization, X-Requested-With");
31  ?>
32

```

### 2. Requêtes Préparées :

- Utilisation de requêtes préparées pour toutes les opérations de base de données afin d'éviter les injections SQL.

```

1  <?php
2
3  include __DIR__ . '\db_connexion.php';
4
5  try {
6      if (
7          $_SERVER['REQUEST_METHOD'] === 'POST' &&
8          isset($_POST['nom']) &&
9          isset($_POST['description']) &&
10         isset($_FILES['image']) &&
11         isset($_POST['continent_id'])
12     ) {
13         $nom = $_POST['nom'];
14         $description = $_POST['description'];
15         $image = file_get_contents($_FILES['image']['tmp_name']);
16         $commentaire = $_POST['commentaire_habitat'] ?? null;
17         $continent_id = $_POST['continent_id'];
18
19         $stmt = $pdo->prepare(
20             "INSERT INTO habitat (nom, description, image, commentaire_habitat, continent_id)
21             VALUES (:nom, :description, :image, :commentaire, :continent_id)"
22         );
23         $stmt->bindParam(':nom', $nom);
24         $stmt->bindParam(':description', $description);
25         $stmt->bindParam(':image', $image, PDO::PARAM_LOB);
26         $stmt->bindParam(':commentaire', $commentaire);
27         $stmt->bindParam(':continent_id', $continent_id, PDO::PARAM_INT);
28
29         if ($stmt->execute()) {
30             echo json_encode(["message" => "L'habitat a été ajouté avec succès."]);
31             error_log("Habitat ajouté avec succès: " . json_encode($_POST));
32         } else {
33             echo json_encode(["message" => "Une erreur s'est produite lors de l'ajout de l'habitat."]);
34             error_log("Erreur lors de l'ajout de l'habitat: " . json_encode($stmt->errorInfo()));
35         }
36     } else {
37         echo json_encode(["message" => "Invalid request."]);
38         error_log("Requête invalide: " . json_encode($_POST) . " Fichiers: " . json_encode($_FILES));
39     }
40 } catch (PDOException $e) {
41     echo json_encode(["message" => "Erreur: " . $e->getMessage()]);
42     error_log("PDOException: " . $e->getMessage());
43 }

```

### 3. Gestion des Erreurs :

- a. Configuration de l'affichage des erreurs de manière appropriée pour éviter la divulgation d'informations sensibles en production.

```

1  <?php
2  ini_set('memory_limit', '256M');
3
4  error_reporting(E_ALL);
5  ini_set('display_errors', 1);
6
7
8  $host = '127.0.0.1';
9  $db = 'arcadia';
10 $user = 'user1';
11 $pass = 'zUWr/KrCIksGxpy9';
12 $charset = 'utf8mb4';
13

```

## Sécurité à venir :

### 1. Contrôles d'Accès :

- Mise en place de mécanismes de contrôle d'accès pour s'assurer que seules les personnes autorisées peuvent accéder aux fonctionnalités ou données sensibles.

### 2. Utilisation de HTTPS :

- Sur un serveur en ligne, utilisation de HTTPS pour sécuriser les communications entre le client et le serveur.

### 3. Protection CSRF (Cross-Site Request Forgery) :

- J'utiliserai des tokens CSRF pour sécuriser les requêtes POST. Angular HttpClient peut être configuré pour inclure des tokens CSRF avec chaque requête.

### 4. Journalisation et Surveillance :

- Mise en place de mécanismes de journalisation pour surveiller les activités suspectes et identifier les tentatives d'intrusion.

### 5. Désactiver le Debug en Production

- Je m'assurerai de désactiver les outils de debug en production en utilisant le mode de production d'Angular.

### 6. Content Security Policy (CSP)

- Je vais implémenter une CSP pour empêcher l'exécution de scripts non autorisés.

### 7. Authentification et Autorisation

- J'utiliserai des services d'authentification robustes. Je m'assurerai que les rôles et permissions sont correctement gérés.

### 8. Gestion des Sessions

- Si j'utilise des sessions, je m'assurerai qu'elles expirent après une période d'inactivité et j'utiliserai des cookies sécurisés (`Secure` et `HttpOnly`).

### 9. Contrôle d'Accès en Fonction des Rôles

- J'implémenterai un contrôle d'accès basé sur les rôles (RBAC) pour limiter l'accès à certaines parties de l'application.

2-4°) Décrivez une veille technologique que vous avez effectuée, sur les vulnérabilités de sécurité.

Récemment, j'ai effectué une veille technologique approfondie sur les vulnérabilités de sécurité pour assurer la robustesse et la protection de mes applications web. Voici comment j'ai procédé :

## 1. Objectif de la Veille

Mon objectif principal était de comprendre les dernières vulnérabilités de sécurité affectant les applications web, en particulier celles construites avec Angular et PHP, et de mettre en place des mesures pour les atténuer.

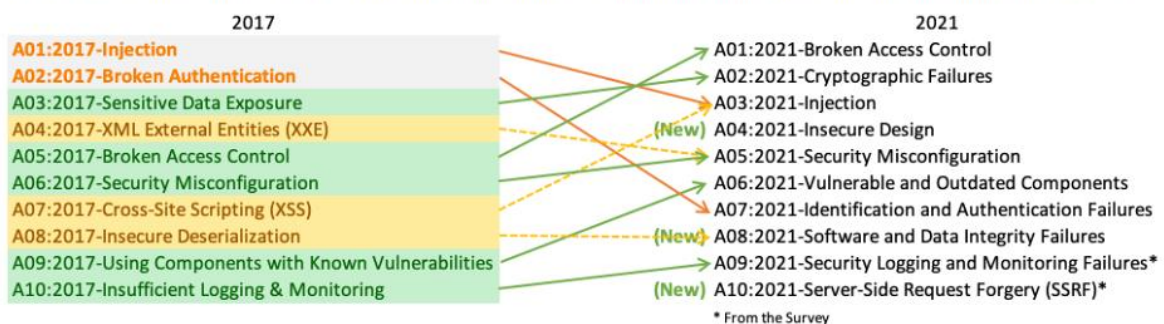
## 2. Sources Utilisées

Pour effectuer cette veille, j'ai utilisé diverses sources fiables et régulièrement mises à jour :

- **OWASP (Open Web Application Security Project)** : Le site OWASP fournit des informations détaillées sur les vulnérabilités courantes et émergentes, ainsi que des recommandations de sécurité.

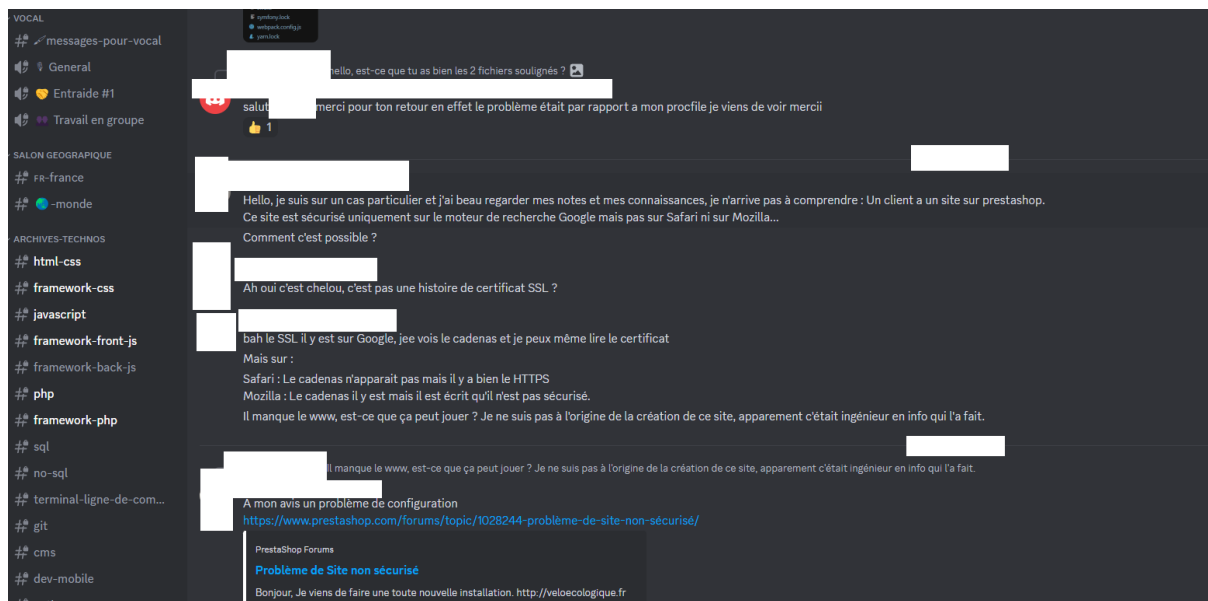
### Top 10 Web Application Security Risks

There are three new categories, four categories with naming and scoping changes, and some consolidation in the Top 10 for 2021.



- **Discord** : Des serveurs en programmation informatique avec d'autres programmeurs





- **Avec des collègues de travail** : J'échange régulièrement avec des collègues de travail notamment sur checkmarx et également sur les chaînes CICD. J'approfondis mes compétences avec ces échanges. J'ai de la chance d'avoir des collègues très au point sur ces sujets.
- **Information de la DSI** : On a des points mensuels avec notre direction de Sécurité Informatique. C'est surtout le techLeade qui voit ça avec eux mais je traîne un peu l'oreille parfois.

### 3. Identification des Vulnérabilités

Pendant ma veille, j'ai identifié plusieurs vulnérabilités critiques :

- **Cross-Site Scripting (XSS)** : Vulnérabilités permettant à un attaquant d'injecter des scripts malveillants dans des pages web.
- **Injection SQL** : Attaques permettant l'exécution de requêtes SQL non autorisées via des entrées utilisateur non sécurisées.
- **Cross-Site Request Forgery (CSRF)** : Attaques exploitant la confiance d'un site dans le navigateur d'un utilisateur.
- **Vulnérabilités des dépendances** : Utilisation de bibliothèques et packages avec des vulnérabilités connues.

### 4. Mesures de Sécurité

À la suite de cette veille, j'ai mis en place plusieurs mesures pour renforcer la sécurité de mon applications :

- **Validation et Sanitisation des Entrées** : Utilisation des mécanismes de validation d'Angular et de PHP pour nettoyer les entrées utilisateur.
- **Requêtes Préparées avec PDO** : Pour prévenir les injections SQL, j'ai systématiquement utilisé des requêtes préparées.

- **Tokens CSRF** : Mise en place de tokens CSRF pour sécuriser les requêtes POST dans mes applications Angular et PHP.
- **Audit des Dépendances** : Utilisation d'outils comme **npm audit** et **Snyk** pour scanner les dépendances et mettre à jour les packages vulnérables.
- **Content Security Policy (CSP)** : Mise en œuvre de CSP pour prévenir les attaques XSS.
- **Formation et Sensibilisation** : Participation à des webinaires et des formations en ligne pour me tenir à jour sur les meilleures pratiques en matière de sécurité.

## 5. Résultats

Grâce à cette veille technologique, j'ai pu identifier et corriger plusieurs vulnérabilités potentielles dans mes applications. J'ai également acquis une meilleure compréhension des menaces actuelles et des méthodes de protection, ce qui m'a permis d'améliorer globalement la sécurité de mes projets.

En conclusion, cette veille technologique sur les vulnérabilités de sécurité m'a permis de renforcer les défenses de mes applications et de garantir une meilleure protection des données et des utilisateurs.

## Partie 3°) recherches

3-1°) Décrivez une situation de travail ayant nécessité une recherche durant le projet à partir de site anglophone. N'oubliez pas de citer la source

Je me suis sourcé sur **stackOverFlow** lorsque je rencontrais des problèmes. C'est essentiellement rédigé en anglais. J'utilise accessoirement google translate pour m'assurer d'avoir bien compris ce que j'ai lus.

3-2°) Mentionnez l'extrait du site anglophone qui vous a aidé dans la question précédente en effectuant une traduction en français.

## Partie 4 : Informations complémentaire

### 4-1°) Autres ressources

Cours développeur

ECF

Examen

Enoncé

Architecture de l'application

organisation des dossiers

BDD

Relationnelle

Charte Graphique

Couleurs primaires

Margin Padding

Border

texte

Images

Logo

La maquette

User Stories

User Stories 1 - Page d'accueil

User Stories 2 - Menu d'application

User Stories 3 - Services

User Stories 4 - Les habitats

User Stories 5 - Avis

User Stories 6 - Espace administra...

User Stories 7 - Espace employé

User Stories 8 - Espace Vétérinaire

User Stories 9 - Connexion

User Stories 10 - Contact

User Stories 11 - Statistique

Persona

tous les personas

Vétérinaires

lundi 20 mai 2024 08:04

Champs d'actions :

- Gestion des comptes rendus** : Remplir des comptes rendus pour chaque animal visité, incluant l'état de l'animal, la nourriture proposée, le grammage, la date de passage, et des détails facultatifs.
- Commentaires sur les habitats** : Donner des avis sur l'état des habitats et suggérer des améliorations si nécessaire.
- Suivi de l'alimentation** : Voir les informations sur la nourriture donnée aux animaux, enregistrées par les employés.

### 4-2°) Informations complémentaires