Journal de développement – Galerie Oselo

Jour 1 – Création de la base de données

- **Tâche** : Création de la base de données galerie_oselo avec deux tables : artworks (œuvres) et warehouses (entrepôts).
- **Progrès** : Écriture du script SQL contenant les champs id, title, artist_name pour les œuvres, et name, address pour les entrepôts.
- **Problèmes** : Oubli de l'attribut AUTO_INCREMENT sur les champs id, ce qui empêchait l'incrémentation automatique.
- **Solution** : Ajout de AUTO_INCREMENT après une recherche sur la documentation MySQL.

Jour 2 – Connexion à la base de données

- **Tâche** : Création du fichier db_connect.php pour établir la connexion avec PDO.
- **Progrès** : Connexion réussie avec $pdo = new PDO(...).
- **Problèmes** : Message d'erreur "Access denied for user 'root'" au début.
- **Solution** : Vérification du mot de passe dans XAMPP (il était vide par défaut) puis correction dans le fichier de connexion.

Jour 3 – Page des œuvres (artworks.php)

- **Tâche** : Développement de la page permettant d'ajouter, modifier, supprimer et assigner des œuvres.
- **Progrès** : Création d'un formulaire d'ajout et d'un tableau d'affichage. Les requêtes SQL fonctionnent.
- **Problèmes** : L'assignation d'une œuvre à un entrepôt ne mettait pas à jour correctement.
- **Solution** : Correction de la requête UPDATE en ajoutant le champ warehouse_id dans la table artworks et test avec un identifiant valide.

Jour 4 – Page des entrepôts (warehouses.php)

- **Tâche** : Mise en place de la gestion des entrepôts (ajout, modification, suppression).
- **Progrès** : Affichage et gestion des entrepôts fonctionnels. Ajout d'une vérification pour éviter les doublons sur le nom.
- **Problèmes** : La suppression ne fonctionnait pas, l'identifiant n'était pas récupéré.
- **Solution** : Modification de filter_input(INPUT_POST, 'id') en filter_input(INPUT_GET, 'id') après avoir constaté que la suppression passait par l'URL.

Jour 5 – Sécurité et tests

- **Tâche** : Mise en place des protections contre les failles XSS et SQL, et réalisation de tests.
- **Progrès** : Utilisation de htmlspecialchars() pour protéger les entrées et de requêtes préparées avec PDO.
- **Problèmes** : Doute sur l’efficacité des protections mises en place.
- **Solution** : Tests manuels réalisés avec du code malveillant (balise <script> et injection SQL). Comportement attendu confirmé.

Jour 6 – Documentation et veille

- **Tâche** : Rédaction de la documentation (README.md), ajout des scénarios de tests et définition d’un plan de veille.
- **Progrès** : Liste des tests fonctionnels et de sécurité intégrée à README.md et tests.md. Plan de veille rédigé pour le suivi de PHP, MySQL et des tendances UI.
- **Problèmes** : Incertitude sur le contenu à inclure dans la partie veille.
- **Solution** : Décision de suivre les mises à jour via les sites officiels (PHP.net, W3Schools) à la suite de recherches sur la veille technologique.

1. Le jeu d’essai fonctionnel est complet

- Dans artworks.php et warehouses.php, les commentaires indiquent que les fonctions (ajouter, modifier, supprimer, assigner) ont été testées.
- Dans README.md et tests.md, j’ai une liste de tests fonctionnels (ex. "Ajouter une œuvre avec le titre 'Mona Lisa' → Apparaît dans le tableau").

2. Les tests de sécurité sont réalisés

- Dans artworks.php et warehouses.php, j’ai utilisé **htmlspecialchars()** pour éviter les XSS et des requêtes préparées pour éviter les injections SQL. J’ai aussi ajouté une limite de longueur pour title et name.
- Dans db_connect.php, PDO est sécurisé.
- Dans README.md et tests.md, j’ai des tests de sécurité (ex. "**Testé XSS** : **<script>** → **S’affiche comme texte**").

- "J’ai testé XSS et injection SQL comme dans tests.md. Les protections (**htmlspecialchars** et **PDO**) marchent. Voir le code dans artworks.php, warehouses.php, et db_connect.php."

3. Le système de veille permet de suivre les évolutions technologiques et les problématiques de sécurité en lien avec les interfaces utilisateur

- Dans README.md, j'ai une section "**Monitoring Plan**" (ex. "Vérifier PHP.net et MySQL.com tous les mois").
- Dans artworks.php et warehouses.php, des commentaires disent que le code est facile à mettre à jour (ex. "**Monitoring: Easy to update form fields**").

- "Je vais vérifier les mises à jour de PHP et MySQL tous les mois sur leurs sites officiels. Si une nouvelle version sort, je mettrai à jour **db_connect.php.** Pour l'interface, je regarderai **W3Schools** pour des idées et adapterai mon code."
- "Mon plan de veille est dans README.md. Le code est prêt à évoluer, voir les commentaires 'Monitoring' dans artworks.php et warehouses.php."

Recherches sur internet – Galerie Oselo

Résolution de problèmes

1. Champ auto-incrémenté dans MySQL

- **Recherche** : "MySQL auto increment"
- **Lien** : https://www.w3schools.com/sql/sql_autoincrement.asp
- **Résultat** : J'ai appris à utiliser AUTO_INCREMENT pour générer automatiquement les identifiants. J'ai corrigé mon script SQL avec la syntaxe id INT AUTO_INCREMENT PRIMARY KEY.

2. Erreur de connexion PDO "Access denied"

- **Recherche** : "PDO access denied for user root"
- **Lien** : https://stackoverflow.com/questions/21646762/pdoexception-sqlstatehy000-1045-access-denied-for-user
- **Résultat** : Le mot de passe par défaut était vide dans XAMPP. J'ai modifié mon fichier db_connect.php avec $password = "", et la connexion a été établie avec succès.

3. Prévention des injections SQL

- **Recherche** : "PHP PDO prevent SQL injection"
- **Lien** : https://www.php.net/manual/fr/pdo.prepared-statements.php

- **Résultat** : J’ai sécurisé mes requêtes avec des instructions préparées via PDO, comme $stmt->execute([$title, ...]) dans artworks.php et warehouses.php.

4. Prévention des attaques XSS

- **Recherche** : "PHP prevent XSS"
- **Lien** : https://www.w3schools.com/php/func_string_htmlspecialchars.asp
- **Résultat** : J’ai utilisé htmlspecialchars() sur toutes les sorties HTML pour empêcher l’exécution de balises script malveillantes.

5. Suppression d’entrepôt – ID non détecté

- **Recherche** : "PHP filter_input GET vs POST"
- **Lien** : https://www.php.net/manual/fr/function.filter-input.php
- **Résultat** : J’utilisais INPUT_POST au lieu de INPUT_GET pour récupérer l’identifiant passé dans l’URL. Après correction, la suppression fonctionne correctement.

Veille technique

1. Dernières versions de PHP

- **Recherche** : "PHP latest version 2025"
- **Lien** : https://www.php.net/
- **Résultat** : PHP 8.2 est la version stable en 2025. Je prévois de vérifier chaque mois les nouvelles versions afin de maintenir la compatibilité dans db_connect.php.

2. Tendances UI en 2025

- **Recherche** : "UI trends 2025"
- **Lien** : https://www.w3schools.com/
- **Résultat** : De nouvelles pratiques apparaissent, comme l’utilisation de modales fluides et de formulaires minimalistes. Je compte m’en inspirer pour améliorer l’interface de artworks.php.

3. Sécurité des interfaces web

- **Recherche** : "Web security UI 2025"
- **Lien** : https://owasp.org/
- **Résultat** : OWASP met l’accent sur les failles XSS, CSRF et injections. Mon application est déjà protégée contre les XSS et les injections SQL. Je continuerai à suivre leurs recommandations.