TP Maintenance

mardi 27 janvier 2026 09:42

Reprise du projet UGSELWEB

Voici une première anayse du projet relevant les problèmes qu'il comporte :

Problème globaux :
- Tout est centralisé dans un fichier
- Base de données individuelles pour les utilisateurs
- Pavé illisible, aucune documentation
- Encodage en latin1 au lieu d'UTF-8 -> mélange par moment
- Utilisation d'un trop vieux moteur de gestion des données (MyISAM)
- Chemins codés en dur
- Pas de gestion de dépendances

Problème de code :
- Trop de variables globales
- Problèmes de caractères spéciaux très récurrent
- Fonctions trop longues et complexes complexité algo qui explose (violation du principe de responsabilité unique)
- Utilisation de {} au lieu de crochet dans des appels de listes ce qui créée un grand nombre d'erreur (actuellement le code en détecte une mais à chaque résolution la suivante est prise en compte)
- Fonctions sql dépréciés de nos jours
- Gestion d'erreurs inexistantes
- Chemins codés en dur
- Pas d'otienté objet que du procédural
- Requêtes sql redondantes
- Nommage de variables incohérent et variable inutilisées

Problèmes de sécurité
- Crédentials exposés en clair
- Injections SQL massives
- Aucune protection contre les injections XSS
- Gestion des sessions non sécurisées
- Upload de fichier sans protection

Bilan d'analyse

En classifiant tous les problèmes, le plus urgent est de migrer la base de données vers MySQLi/PDO en centralisant toutes les données dans une seule base, retirer les crédentials du code et corriger les différentes failles XSS/SQL. Suite à cela il va falloir migrer le projet vers UTF-8, le restructurer avec un outils moderne comme Symfony et implémenter une vraie gestion des erreurs. Enfin, si le temps le permet il faudra ajouter des tests unitaires, écrire une documentation complète et proposer une amélioration graphique du projet.

Pour résoudre cette problématique, je propose d'utiliser les framework Symfony 7.6 pour le back-end et Angular 21 pour le front-end. Pour le système de données une base en mariadb géré avec phpmyadmin me paraît envisageable.
Avec symfony on a une architecture MVC qui est idéale pour coder en orienté objet et séparer correctement les différents aspects de l'application. De plus, séparer le front-end et le back-end

dans deux sous projets différents permet d'éviter que les équipes de back et de front se marchent dessus lors d'une reprise future du projet. Le choix des outils Symfony et Angular me paraissent pertinent car il sont très populaire dans l'industrie. De plus, ils sont complets et bien documentés ce qui permet une compréhension simplifiée du code dans le cas où la documentation classique ne suffit pas. Enfin, pour le back-end utiliser Symfony permet de travailler avec composer et ainsi intégrer une mise à jour régulière des dépendances utilisées dans le projet et éviter de nouvelles failles liées à celles-ci.
Pour ce qui est des tests et de l'intégration continue, je préconise l'utilisation de Jenkins couplé à Sonarqube. Cela permet à chaque mise en pré-production ou production d'analyser complètement le code avec Sonarqube puis d'exécuter automatiquement des tests unitaires avec Jenkins. Il paraît également envisageable de déployer la pré-production sur un serveur avec une visibilité réduite comme sur Alwaysdata pour identifier certains problèmes rencontrables lors d'une mise en production réelle.

Modulation d'évaluation du Projet :

R6, maintenance, petite grille pour vous aider dans l'évaluation de votre projet :
1 : inexistant
2 : mauvais
3 : moyen
4 : bon
5 : le graal

Choix des techno, pertinence:
1 à 5 Principes
SOLID respectés : Oui / Non
Loi de Demeter : OUI / NON
KISS : 1 à 5
Documentation : oui / non
qualité de la doc : 1 à 5
Linter : présent oui / non
Aucune erreur 1 à 5
Usage de métriques : oui / non
pertinence de l'implémentation : 1 à 5 (tests réguliers programmés, outils…)
Structure des branches , pertinence : 1 à 5
Présence d'outils de contrôle (Linter…) oui / non

Tests : tests unitaires : oui / non,
pas d'erreur : 1 à 5
behavior : oui / non,
pas d'erreur : 1 à 5
finaux : oui / non,
pas d'erreur : 1 à 5