ISI WEB

Nouha MEQDAD & Lou HONNART-GIDOIN

Documentation technique

Structure du projet (MVC)

- **Modèle (Models)** : gère les données et la logique métier.
    - `Article.php` → gestion des articles (CRUD, slug, status, image à la une).
    - `Commentaire.php` → gestion des commentaires (ajout, modération, notifications).
    - `Tag.php` → gestion des tags et associations articles-tags.
    - `User.php` → gestion des utilisateurs, rôles et permissions.
- **Contrôleurs (Controllers)** : reçoivent les requêtes, appellent les modèles et choisissent la vue à afficher.
    - `BlogController.php` → gestion de la partie publique (affichage articles, détail, ajout de commentaires).
    - `AdminController.php` → gestion de l'administration (articles, commentaires, utilisateurs, tags).
    - `AuthController.php` → connexion et déconnexion des utilisateurs.
- **Vues (Views)** : templates Twig qui affichent le contenu à l'utilisateur.
    - `/Views/blog` → pages publiques (`home.twig`, `article.twig`, `login.twig`).
    - `/Views/admin` → interface d'administration (`dashboard.twig`, `commentaires.twig`, `tags.twig`, `users.twig`, etc.).
    - `nav.twig` → menu commun.

- **Config** : fichiers de configuration.
    - `Database.php` → connexion à la base de données avec PDO (Singleton).
- **Public** : fichiers accessibles depuis le navigateur.
    - `index.php` → point d'entrée du site et routeur central.
    - `/uploads` → images ou fichiers uploadés.

Design Patterns utilisés

I. **MVC (Modèle-Vue-Contrôleur)** :

A. Sépare clairement la logique métier (Models), le contrôle des requêtes (Controllers) et l'affichage (Views).

II. **Singleton** :

A. Utilisé pour la classe `Database.php`.

B. Garantit qu'une seule instance de connexion PDO existe dans toute l'application, évitant les connexions multiples.

III. **POO (Programmation Orientée Objet)** :

A. Utilisation de classes pour représenter les entités (Articles, Commentaires, Utilisateurs, Tags).

B. Méthodes statiques pour certaines opérations comme `create`, `update`, `delete`.

Répartition du travail au sein du binôme

1) NOUHA MEQDAD

- Architecture du projet(MVC,PHP OOP,singleton,bases de données)
- Gestion des articles
- Gestion des commentaires
- Harmonisation de l'affichage entre la partie publique et l'administration

2) Lou

- Front-end
- Gestion connection
- gestion rôles et permission utilisateur
- gestion tag

Difficulté :

Nous avons rencontré des difficultés principalement sur l'implémentation des contraintes d'accès selon les rôles, ce qui a nécessité d'adapter en partie l'architecture initiale. La gestion des formulaires POST, des redirections et des actions multiples, ainsi que la synchronisation entre la base de données et l'affichage via Twig, ont également été complexes, notamment pour la modération des commentaires. Il a fallu organiser le projet de manière lisible et maintenable et décider de ne considérer qu'un seul rôle à la fois de manière hiérarchique pour simplifier la logique.

Chose à tirer du projet :

Ainsi ce projet nous a appris l'importance de repérer et traiter en priorité à la racine du projet, les fonctionnalités susceptibles d'influencer la structure globale du code