Documentation Technique du Projet Blog

1. Architecture Générale : MVC

A. Modèle (Model)

- **Rôle :** Gère les données, la logique métier et les interactions avec la base de données.
- **Emplacement :** app/modeles/
- **Fichiers Clés :**
    - Articles.php : Gère le CRUD des articles (création, lecture, mise à jour).
    - Connexion.php : Gère l'authentification (Login/Register) et les transactions.
    - Dashboard.php : Contient les requêtes complexes d'administration (KPIs, gestion utilisateurs/rôles).
    - Permissions.php : Gère le système RBAC (Role-Based Access Control).

B. Vue (View)

- **Rôle :** Gère l'interface utilisateur et l'affichage des données.
- **Moteur de Template : Twig**.
- **Emplacement :** app/vues/
- **Fichiers Clés :**
    - base.twig : Le layout principal (Header/Footer) étendu par les autres vues.
    - dashboard.twig : La vue modulaire qui inclut dynamiquement des sous-vues (gestionUtilisateur.twig, mesArticles.twig) selon les permissions.
    - creer_article.twig : Formulaire unique pour la création et l'édition.

C. Contrôleur (Controller)

- **Rôle :** Reçoit la requête, interroge le Modèle, et envoie les données à la Vue.
- **Emplacement :** app/controlleurs/
- **Fichiers Clés :**
    - ConnexionControlleur.php : Orchestre le tableau de bord et la gestion des utilisateurs.
    - ArticleControlleur.php : Gère l'affichage, la création et l'édition des articles.

---

2. Point d'Entrée : Le Front Controller

Le projet utilise un **Front Controller** unique situé dans public/index.php.

- **Fonctionnement :** Toutes les requêtes HTTP passent par ce fichier.
- **Routage :** Un switch($uri) analyse l'URL et dirige vers le contrôleur et la méthode appropriés (ex: /creer-article appelle $ArticleControlleur->creer()).
- **Gestion des erreurs :** Si aucune route ne correspond, le bloc default tente de charger un article par ID ou renvoie une 404.

---

3. Design Patterns Utilisés

Le projet implémente plusieurs patrons de conception pour assurer la robustesse et la maintenabilité du code.

A. Singleton

Le pattern Singleton est utilisé pour garantir qu'une classe n'a qu'une seule instance et fournir un point d'accès global.

- **Database :** Database::getInstance()->getConnection() assure une seule connexion SQL active par requête.
- **SessionManager :** SessionManager::getInstance() permet de manipuler la session ($_SESSION) de manière orientée objet sans conflits.
- **Logger :** Logger::getInstance() centralise l'écriture des logs.

B. Dependency Injection (Injection de Dépendances)

Les dépendances externes sont injectées dans les classes plutôt que créées à l'intérieur.

- **Exemple :** Le moteur de template Twig\Environment est injecté via le constructeur des contrôleurs.

C. Data Access Object (DAO)

Les modèles agissent comme des DAO, encapsulant les requêtes SQL complexes pour que les contrôleurs ne manipulent que des objets ou des tableaux de données.

- **Exemple :** Dashboard::getUtilisateursAvecRoles() masque la complexité des jointures SQL (JOIN) au contrôleur.

---

4. Gestion de la Sécurité et des Données

A. Transactions SQL (Atomicité)

Le principe du "Tout ou Rien" est appliqué lors des opérations critiques impliquant plusieurs tables.

- **Inscription :** Lors de la création d'un compte, l'insertion dans Utilisateurs et l'attribution du rôle dans Role_User sont encapsulées dans beginTransaction() et commit(). En cas d'erreur, rollBack() annule tout.
- **Suppression :** La suppression d'un utilisateur supprime d'abord ses articles puis son compte dans une transaction.

B. RBAC (Role-Based Access Control)

La sécurité n'est pas basée uniquement sur des rôles, mais sur des **Permissions** granulaires.

- **Vérification :** Le contrôleur vérifie une capacité spécifique (ex: article_creer, article_publier) plutôt qu'un rôle global.

C. Sécurité des Mots de Passe

- Hachage : Utilisation de password_hash($password, PASSWORD_DEFAULT) à l'inscription.
- Vérification : Utilisation de password_verify() à la connexion.

---

5. Stack Technique

- **Langage :** PHP 8+ (Typage fort, Attributs).
- **Base de données :** MySQL / MariaDB.
- **Moteur de Template :** Twig.
- **Frontend :**
    - **Bootstrap 5** (Structure et composants CSS).
    - **Alpine.js** (Interactivité légère : Modals, dropdowns notifications).
    - **EasyMDE** (Éditeur Markdown WYSIWYG).
- **Gestionnaire de paquets :** Composer (Autoloading PSR-4).