Documentation Technique : Projet 3A2526-BLOG
PERRONNET Arthur

1. Présentation de l'Architecture

Le projet repose sur une architecture MVC (Modèle-Vue-Contrôleur) personnalisée. Cette structure permet une séparation stricte entre la logique métier, la gestion des données et l'interface utilisateur, facilitant de cette manière la maintenance et l'évolutivité.

Points clés :
- Point d'entrée unique : Toutes les requêtes passent par *public/index.php*.
- Encapsulation : Le dossier *app/* contient toute la logique protégée, non accessible par les utilisateur via leur navigateur.
- Standardisation : Utilisation de classes de base pour l'héritage.

2. Structure du MVC

A. Les Contrôleurs (*app/Controllers*)

Ils agissent comme des intermédiaires. Ils reçoivent les instructions de l'utilisateur, interrogent les modèles et choisissent la vue à afficher.
- *BaseController.php* : La classe parente dont héritent tous les autres contrôleurs. Elle contient la méthode de rendu (**render()**).
- *HomeController.php* : Gère l'affichage des pages principales du site.
- *LoginController.php* : Gère l'authentification (connexion, déconnexion, dashboard administrateur).
- *PostController.php* : Gère le CRUD (Création, Lecture, Mise à jour, Suppression) des articles du blog.

B. Les Modèles (*app/Models*)

Ils sont responsables de la manipulation des données via SQL.

- *BaseModel.php* : Classe parente qui donne accès à la base de données.
- *AccountModel.php* : Gestion de la table des utilisateurs.
- *PostModel.php* : Gestion des articles et de leur contenu.
- *CommentsModel.php* : Gestion des commentaires liés aux articles.
- *PermissionsModel.php* : Gestion des rôles et droits d'accès (ACL).

C. Les Vues (*app/Views*)

Contient les fichiers de templates (Twig). C'est ce que les utilisateurs verront en accédant au site.

3. Le noyau et Design Patterns (*app/Core*)

Le dossier Core contient les composants bas niveau et les moteurs de l'application.

Design Patterns (Patrons de Conception) :

1. Singleton (via *Database.php*) : Ce pattern garantit qu'une seule instance de la connexion à la base de données est créée pendant l'exécution d'un script, évitant ainsi de saturer le serveur de base de données.
2. Dependency Injection (via *Logger.php* et *SessionManager.php*) : Ces services sont conçus pour être injectés dans les contrôleurs. SessionManager centralise la gestion des variables $_SESSION de manière sécurisée, tandis que Logger écrit les événements dans *logs/app.log.*
3. Template Method (Héritage) : En utilisant des classes BaseController et BaseModel, le projet impose une structure commune à tous les composants enfants.

4. Dossiers Système et Configuration

- *public/* :
    - *index.php* : Le contrôleur frontal (Front Controller). Il charge l'Autoloader, initialise la session et lance le routage.
    - *.htaccess* : Configure la réécriture d'URL pour rediriger tous les flux vers index.php.
    - *assets/* : Contient les ressources statiques (Images, CSS, JS).
- *logs/* : Stocke les journaux d'erreurs et d'activités (*app.log*). Très utile pour le debug lors de la production.
- *vendor/* : Dossier géré par Composer contenant les bibliothèques tierces.

5. Cycle de vie d'une requête

1. Requête : L'utilisateur demande monblog.fr/post/show/1.
2. Routage : *index.php* interprète l'URL et instancie PostController.
3. Action : La méthode show(1) du contrôleur est appelée.
4. Données : Le contrôleur appelle PostModel pour récupérer l'article n°1.
5. Log : Le Logger peut enregistrer que l'article a été consulté.
6. Réponse : Le contrôleur passe les données à une vue dans app/Views et l'affiche à l'utilisateur.