MARIE-ANDRÉE HEALEY-CÔTÉ

Programmation Web avancée

582-31B-MA, gr.24610

TP 2

Architecture MVC

Travail présenté à

M. Marcos Sanches

Département de Formation continue – Programme NWE.0F

Collège de Maisonneuve

Le 2 novembre 2025

Présentation du projet : Migration vers l'architecture MVC

1. Contexte et objectifs

Ce projet a pour objectif de **migrer une application existante vers une architecture MVC (Modèle-Vue-Contrôleur)** afin d'améliorer la structure du code, la maintenabilité, et la séparation des responsabilités. L'approche MVC permettra de mieux organiser le développement, de faciliter les évolutions futures et de rendre l'application plus modulaire et robuste.

2. Architecture et organisation

L'application sera structurée selon le modèle **MVC** :

- **Modèle (Model)** : gère les interactions avec la base de données et représente les entités du système.
- **Vue (View)** : s'occupe de l'affichage et du rendu des données via le moteur de template **Twig**.
- **Contrôleur (Controller)** : traite les requêtes, appelle les modèles, et envoie les données aux vues.

Une gestion claire des routes sera mise en place pour faire le lien entre les URL et les contrôleurs.

3. Développement

3.1. Routage

- Création d'un système de **routes** pour gérer la navigation.
- Chaque route sera associée à un **contrôleur** et une **méthode d'action** spécifique.
- Exemple :
    - / → HomeController::index()
    - /livres → LivreController::index()
    - /livre/create → LivreController::create()
    - /livre/edit/{id} → LivreController::edit($id)
    - /Livre/delete/{id} → LivreController::delete($id)

3.2. Contrôleurs

- Chaque fonctionnalité aura son **propre contrôleur** (ex. : UserController, LivreController, HomeController).
- Les contrôleurs appelleront les modèles et transmettront les données aux vues.

3.3. Modèles

- Création de **modèles** pour représenter les tables de la base de données (ex. : Auteur, Article, Categorie, Editeur).
- Chaque modèle contiendra les opérations CRUD nécessaires :
    - create(), read(), update(), delete()
- Une couche d'accès aux données sera mise en place avec PDO et des requêtes préparées pour la sécurité.

3.4. Vues

- Développement de **quatre vues principales** :
    1. **Page d'accueil** (vue statique)
    2. **Vue de liste** des éléments (ex. : liste des articles)
    3. **Vue de création/modification** d'un élément
    4. **Vue de détail** ou de suppression
- Les vues seront rendues à l'aide du moteur de templates **Twig** pour une séparation claire du code PHP et du HTML.

3.5. Validation et gestion des erreurs

- Mise en place de la **validation des données** côté serveur avant insertion ou modification.
- Gestion des erreurs (404, 500, champs invalides, etc.) avec des pages d'erreur dédiées.

3.6. Design et ergonomie

- Intégration d'une **feuille de style CSS** pour une présentation soignée et responsive.
- Navigation claire entre les différentes sections de l'application.

4. Résultats attendus

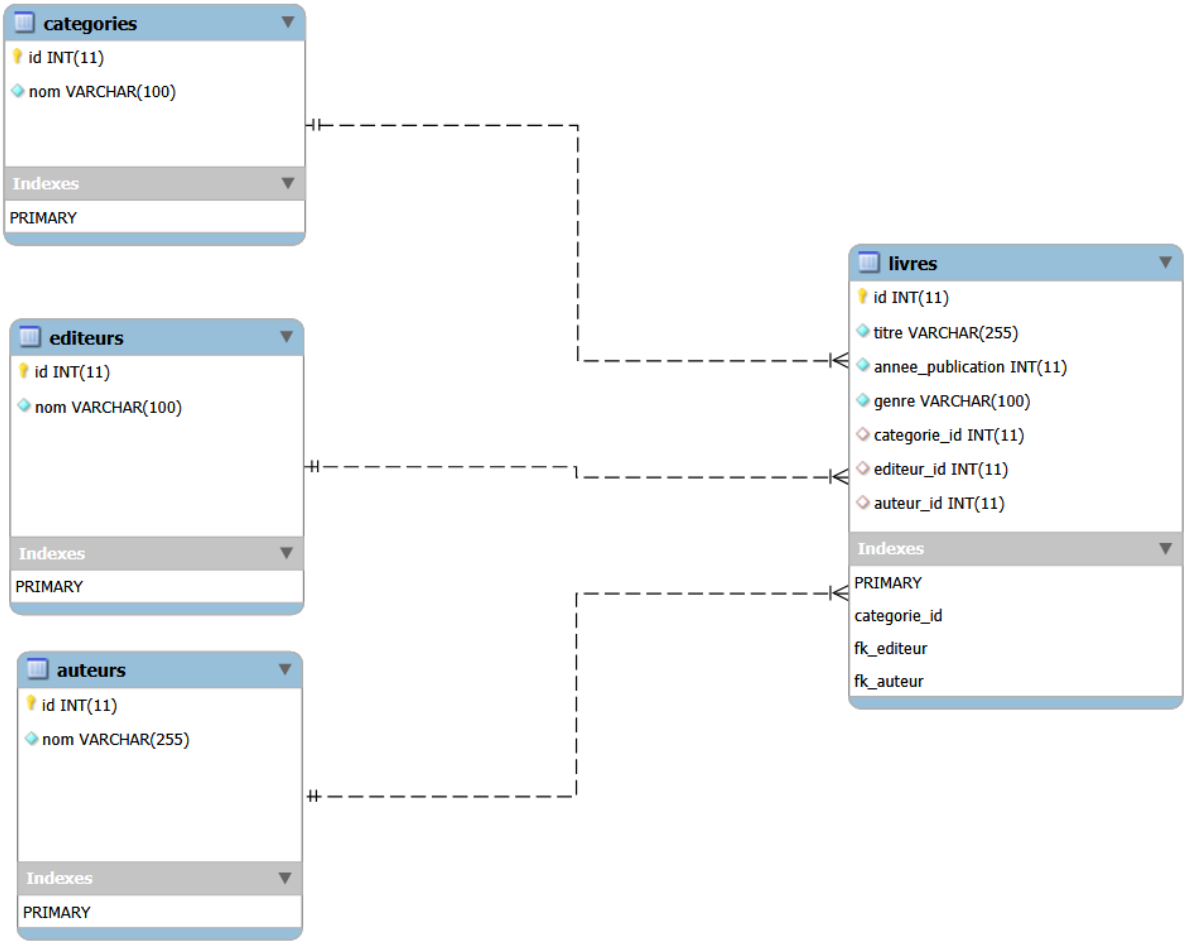
À la fin du projet, l'application devra :

- Être **fonctionnelle** et **navigable** ;
- Respecter la structure **MVC** ;
- Permettre d'exécuter des opérations **CRUD complètes** sur une table représentative ;
- Être **sécurisée** (validation, requêtes préparées) ;
- Offrir une **interface claire et homogène** grâce à Twig et au CSS intégré.

5. Conclusion

Ce projet de migration vers l'architecture MVC permettra de renforcer la qualité du code, d'améliorer la lisibilité et la maintenance du projet, tout en respectant les bonnes pratiques de développement web moderne.

Structure de la base de données (ERD)



Lien GitHub

Lien : https://github.com/marieAndreeHealeyCote/A25-58231BMA-24610