Rapport de Projet - API de Gestion de Livres

1. Introduction

Dans le cadre de ce projet, nous avons développé une API permettant de gérer une base de données de livres. Cette API offre des fonctionnalités CRUD (Create, Read, Update, Delete) pour interagir avec les informations relatives aux livres, telles que le titre, l'auteur, la date de publication, etc. L'objectif principal de cette API est de fournir une interface standardisée et sécurisée pour les applications clientes (sites web, applications mobiles, etc.) afin de manipuler les données des livres de manière efficace et fiable.

2. Stack Technologique

Pour le développement de cette API, nous avons choisi d'utiliser les technologies suivantes :

- **Langage :** PHP, HTML, CSS
- **Framework :** Electron (de Visual Studio Code)
- **Base de données :** MySQL (via phpMyAdmin)

Le choix du langage PHP nous a permis de bénéficier de sa popularité, de sa maturité et de sa vaste communauté de développeurs. L'utilisation de HTML et CSS nous a permis de créer une interface utilisateur attrayante et responsive.

Quant au framework, nous avons sélectionné Electron, qui nous a permis de développer une application de bureau multiplateformes à partir de technologies web (HTML, CSS, JavaScript). Cela nous a offert une grande flexibilité et la possibilité d'intégrer facilement notre API de gestion de livres.

Enfin, la base de données MySQL a été retenue pour sa fiabilité, sa facilité d'utilisation et son excellente intégration avec PHP. L'interface phpMyAdmin nous a également facilité la gestion de la base de données.

3. Conception de l'API

L'API de gestion de livres comprend les endpoints suivants :

3.1. Création d'un livre

Méthode : POST
Endpoint : http://localhost/API-BOOK/api/create.php

Réponse : Un message indiquant que le livre a été ajouté avec succès, avec son nouvel ID.

3.2. Récupération de la liste des livres

Méthode : GET
Endpoint : http://localhost/API-BOOK/api/read.php

Réponse : Un tableau contenant les informations de chaque livre, affiché sous forme de table HTML.

3.3. Mise à jour d'un livre

Méthode : POST
Endpoint : http://localhost/API-BOOK/api/update.php

Réponse : Un message indiquant que le livre a été mis à jour avec succès.

3.4. Suppression d'un livre

Méthode : DELETE
Endpoint : http://localhost/API-BOOK/api/delete.php

Réponse : Un message indiquant que le livre a été supprimé avec succès.

5. Difficultés Rencontrées et Solutions

Au cours du développement de cette API, nous avons rencontré quelques défis techniques :

- **Gestion de la date de publication :** Initialement, nous avions choisi de stocker la date de publication sous forme de chaîne de caractères dans la base de données. Cependant, cela posait des problèmes lors des opérations de tri et de filtrage. Nous avons finalement opté pour le format date natif de MySQL, ce qui a grandement simplifié la manipulation des dates.
- **Sécurité et authentification :** Nous avons ajouté une couche de sécurité pour permettre aux utilisateurs connectés uniquement de pouvoir utiliser delete, create et update (read reste publique); puis update ne s'applique pas à n'importe quel livre. Nous avons donc ajouté un système de session lié à l'ID de la table utilisateur, permettant à un utilisateur connecté d'ajouter puis modifier et/ou supprimer les livres qu'il a ajoutés sans pouvoir toucher aux autres livres, mais il lui reste possible de les consulter.
- **Requêtes Postman :** La méthode d'authentification empêchait les requêtes de s'exécuter, d'où la nécessité de créer une version sans authentification de l'API se trouvant dans le dossier "api-pst" et les méthodes utilisées dans le code ne permettent que les requêtes de forme "x-www-form-urlencoded" depuis le "body".
- **Requête sur update.php :** Nous devions permettre de pouvoir modifier seulement les attributs voulus sans avoir à toucher ou remplir les cases non voulues, d'où la nécessité d'utiliser la méthode POST et modifier la nature des cases du formulaire (seul "id + au moins 1 attribut" sont nécessaires pour déclencher la modification).
- **Gaspillage d'ID :** Chaque livre ajouté prenait l'ID suivant par implémentation, ce qui signifie qu'un livre supprimé ne verra plus son ID être attribué à un autre livre, un gaspillage se crée. Pour résoudre cela, nous avons ajouté au code de create.php une fonction qui analyse les IDs et récupère la première ID disponible par ordre croissant pour l'attribuer au nouveau livre ajouté.
- **Gestion des erreurs :** Nous avons dû mettre en place une gestion robuste des erreurs, afin de renvoyer des réponses HTTP appropriées et des messages d'erreur clairs aux clients en cas de problème (requête invalide, ressource non trouvée, connexion échouée, etc.).

6. Conclusion

Ce projet de développement d'une API de gestion de livres a été une expérience enrichissante. Nous avons pu mettre en pratique nos compétences en programmation PHP, en conception d'interfaces web avec HTML et CSS, ainsi que notre capacité à utiliser MySQL via phpMyAdmin et à développer une application de bureau avec le framework Electron.

Les principales réalisations de ce projet sont :

- Mise en place d'une API RESTful avec PHP, offrant des endpoints CRUD pour la gestion des livres.
- Intégration de MySQL comme base de données pour le stockage efficace des informations sur les livres.
- Développement d'une interface web et peut-être bientôt une application de bureau multiplateformes avec Electron, permettant d'interagir avec l'API.

Répertoire Github : https://github.com/FPALERA/API-BOOK