Rapport de Projet : Gestion d'une Médiathèque avec Django

Table des matières

Introduction

Le projet consiste à développer une application web de gestion pour une médiathèque. Cette application permet aux bibliothécaires de gérer les emprunts, les retours et les membres, tandis que les membres peuvent consulter les médias disponibles. L'application est construite avec le framework Django et utilise une base de données PostgreSQL pour la persistance des données.

Objectifs du Projet

1. Gérer les membres (ajout, modification, suppression, blocages).
2. Gérer les médias (livres, CDs, DVDs, jeux de plateau).
3. Permettre les emprunts et les retours de médias.
4. Fournir une interface utilisateur claire pour les bibliothécaires et les membres.
5. Implémenter une API REST pour certaines opérations.
6. Assurer la qualité du code avec des tests unitaires.

Architecture du Projet

Structure Django

Le projet suit une structure standard Django avec les composants suivants :

1. Applications Django :

- `management` : contient la logique pour gérer les membres, les emprunts et les médias.

2. Modèles :

- Member : représente les membres de la médiathèque avec des informations comme le nom, l'email, et leur statut (éventuellement bloqué).
- Media (classe abstraite) : classe parente pour les médias spécifiques.
  - Book : spécialisation pour les livres.
  - Cd : spécialisation pour les CDs.

- Dvd : spécialisation pour les DVDs.

- BoardGame : spécialisation pour les jeux de plateau.

- Borrow : gère les emprunts avec des relations vers les membres et les médias.

3. API REST :

- ViewSets et serializers pour exposer des opérations via des endpoints REST.

4. Base de données :

PostgreSQL est utilisée comme SGBD.

5. Tâches Asynchrones :

- Celery est utilisé pour vérifier régulièrement les emprunts en retard et envoyer des notifications.

Fonctionnalités Implémentées

1. Gestion des Membres

- Ajout, modification et suppression de membres via l'interface bibliothécaire.

- Blocage automatique des membres ayant des emprunts en retard.

2. Gestion des Médias

- Ajout de nouveaux médias (livres, CDs, DVDs, jeux de plateau).

- Affichage des médias disponibles et empruntés.

3. Emprunts et Retours

- Gestion des emprunts avec vérification des disponibilités.

- Retour des médias avec mise à jour de leur statut.

4. Notifications Automatiques

- Envoi d’alertes pour les emprunts en retard.

- Mise à jour asynchrone via Celery.

5. Interface Utilisateur

- Templates HTML pour afficher les membres, les médias et les emprunts.

- Utilisation de Bootstrap pour une mise en page responsive.

Qualité du Code

1. Lisibilité et Organisation

- Code bien structuré avec des classes et des méthodes claires.

- Utilisation des bonnes pratiques Django (ORM, ViewSets).

2. Tests Unitaires

- Tests pour les modèles principaux (ég. : `Member`, `Borrow`).

- Couverture partielle des fonctionnalités.

3. Sécurité

- Utilisation de PostgreSQL pour une gestion robuste des données.

- Validation des données en entrée pour éviter les erreurs.

Améliorations Recommandées

1. Tests Complémentaires :

- Ajouter des tests pour les emprunts et les retours.

- Tester la logique des blocages.

2. Documentation :

- Rédiger un fichier README.md pour expliquer comment installer et exécuter le projet.
- Ajouter des commentaires au code pour clarifier certaines sections complexes.

3. Gestion Dynamique des Emprunts :

- Rendre la gestion des emprunts plus flexible en gérant tous les types de médias dynamiquement.

4. Améliorations Visuelles :

- Ajouter plus d'éléments visuels sur l'interface utilisateur (icônes, étiquettes).

Conclusion

Ce projet atteint les objectifs principaux fixés dans le cahier des charges. Les fonctionnalités essentielles sont prêtes, et la structure du code est conforme aux bonnes pratiques Django. Avec quelques ajustements mineurs (tests et documentation), l'application pourrait être utilisée dans un environnement réel.