Logiciel de gestion de médiathèque

Rapport détaillé

sezer dogan

Table des matières

1. Étude et correctifs du code fourni

Le projet a démarré avec un code initial fourni en mode console, incluant uniquement quelques classes modèles rudimentaires. Voici les étapes entreprises pour améliorer ce code :

1.1 Analyse du code existant

- Le code était basique et ne respectait pas les normes Django.

- Les classes manquaient de structuration et de séparation des responsabilités (absence de principes POO).

- Aucune persistance des données n'était prévue.

1.2 Correctifs appliqués

1. Conversion du projet en une application Django.

2. Création de classes modèles conformes aux besoins de la médiathèque :

  - Classe `Media` pour représenter les livres, CD, DVD, etc.

  - Classe `Member` pour gérer les utilisateurs.

  - Mise en place d'un héritage pour les spécificités (par exemple, les jeux de plateau).

3. Séparation des fichiers selon les conventions Django :

  - `models.py` pour les modèles.

  - `views.py` pour la logique métier.

  - `urls.py` pour la gestion des routes.

  - `tests.py` pour les tests unitaires.

4. Migration vers SQLite pour garantir une exécution multiplateforme sans prérequis.

2. Mise en place des fonctionnalités demandées

2.1 Application bibliothécaire

L'application permet de :

- Créer un membre-emprunteur.

- Afficher la liste des membres.

- Mettre à jour un membre.

- Supprimer un membre.

- Afficher la liste des médias.

- Ajouter un média.

- Gérer les emprunts (création, retour).

2.2 Application membre

L'application pour les membres permet uniquement de consulter la liste des médias disponibles.

2.3 Contraintes métiers

Les contraintes suivantes ont été respectées :

- Un membre ne peut pas avoir plus de 3 emprunts simultanés.

- Les emprunts doivent être retournés dans un délai d'une semaine.

- Les membres avec des emprunts en retard sont bloqués.

- Les jeux de plateau ne peuvent pas être empruntés.

3. Stratégie de tests

3.1 Tests unitaires

- Chaque fonctionnalité principale est testée dans le fichier `tests.py`.

- Exemple de cas testés :

  - Création et suppression d'un membre.

  - Gestion des emprunts (limite, retours).

  - Blocage des membres en retard.

Commandes pour exécuter les tests :

python manage.py test

3.2 Jeu d'essais

Un fichier `fixtures.json` est inclus pour précharger des données de test.

Commandes pour charger les données :

python manage.py loaddata fixtures.json

3.3 Tests manuels

Scénarios vérifiés :

- Création, modification, et suppression de membres via l'interface utilisateur.
- Ajout et emprunt de médias.
- Vérification des emprunts en retard.
- Consultation de la liste des médias par les membres.

4. Instructions pour exécuter le projet

Prérequis

- Installer **Python** (version 3.10 ou supérieure).
- Installer **Git**.

Étapes

1. Cloner le dépôt :

```
git clone <URL_DU_DEPOT>
cd PythonProject
```

2. Créer un environnement virtuel et installer les dépendances :

```
python -m venv .venv
source .venv/bin/activate  # Pour Windows : .venv\Scripts\activate
pip install -r requirements.txt
```

3. Appliquer les migrations :

python manage.py makemigrations

python manage.py migrate

4. Charger le jeu d’essais :

python manage.py loaddata fixtures.json

5. Lancer le serveur :

python manage.py runserver

6. Accéder à l’application dans un navigateur :

http://127.0.0.1:8000

5. Base de données et jeu d’essais

La base de données SQLite est incluse par défaut (`db.sqlite3`).

Un fichier `fixtures.json` contient un jeu d’essais permettant de tester rapidement le projet.

Exemple de données dans `fixtures.json` :

```json
[
 {
  "model": "management.member",
  "pk": 1,
  "fields": {
   "name": "John Doe",
   "blocked": false,
   "borrow_count": 0
  }
```

```
  },
  {
    "model": "management.media",
    "pk": 1,
    "fields": {
      "name": "Python Basics",
      "author": "Jane Smith",
      "available": true
    }
  }
]
```

---

6. Conclusion

Le projet respecte les contraintes initiales et est compatible avec Windows, macOS et Linux grâce à l'utilisation de SQLite. Toutes les fonctionnalités demandées ont été implémentées, et un ensemble complet de tests a été réalisé pour valider le fonctionnement du logiciel.