Rapport du travail effectué

DEVOIR CEF POO AVEC PYTHON

ARNIAUD Anthony
N° élève : F464172T

Sommaire

1. Introduction

Ce rapport présente la modernisation du système de gestion d'une médiathèque à travers la transformation d'un script Python rudimentaire en une application web fonctionnelle avec Django.

- Code à reprendre :

```
def menu():
    print("menu")

if __name__ == '__main__':
    menu()


class livre():
    name = ""
    auteur = ""
    dateEmprunt = ""
    disponible = ""
    emprunteur = ""

class dvd():
    name = ""
        realisateur = ""
        dateEmprunt = ""
        disponible = ""
    emprunteur = ""


class cd():
    name = ""
    artiste = ""
    dateEmprunt = ""
    disponible = ""
    emprunteur = ""

class jeuDePlateau :
    name = ""
    createur = ""

class Emprunteur():
    name = ""
        bloque = ""


def menuBibliotheque() :
    print("c'est le menu de l'application des bibliothéquaire")




def menuMembre():
    print("c'est le menu de l'application des membres")
    print("affiche tout")
```

2. Étude et Correctifs du Code Fournis

Code initial fourni :

- Le code initial était un script Python rudimentaire en mode console, non structuré et non fonctionnel.

Problèmes identifiés :

- Absence de bonnes pratiques en programmation orientée objet (POO).
- Classes incomplètes et non fonctionnelles (absence de méthodes).
- Structure non adaptée pour une application web.
- Pas de persistance des données.
- Non-respect des principes Django (aucune utilisation des modèles, vues, templates ou URLs).

Correctifs apportés :

- Transformation du script console en une application web utilisant Django.
- Organisation du projet en deux applications :
    - **bibliothecaire** : Gestion des médias, emprunteurs et emprunts.
    - **membre** : Consultation des médias.

- Mise en place des modèles, vues, URLs, et templates dans chaque application.
- Intégration d'une base de données SQLite pour la persistance des données.
- Implémentation des contraintes métiers dans les modèles et les vues :
    - Limitation des emprunts à 3 par membre.
    - Gestion des emprunts en retard.
    - Interdiction d’emprunter les jeux de plateau.

3. Mise en Place des Fonctionnalités Demandées

Application Bibliothécaire :

- **Créer un membre-emprunteur** : Interface pour ajouter un membre.
- **Afficher la liste des membres** : Statut (bloqué ou actif) visible.
- **Mettre à jour un membre** : Modification des informations.
- **Afficher la liste des médias** : Disponibilité affichée.
- **Créer un emprunt** : Réservation d'un média.
- **Ajouter un média** : Ajout de nouveaux éléments (livres, CDs, DVDs, jeux de plateau).
- **Retourner un emprunt** : Mise à jour des disponibilités.

Application Membre :

- **Afficher la liste des médias** : Consultation des médias disponibles.

Contraintes Métier Implémentées :

- Limitation des emprunts à 3 par membre.
- Blocage des emprunteurs ayant un emprunt en retard.

- Emprunt limité aux types "Livre", "DVD", et "CD" (les jeux de plateau ne peuvent pas être empruntés).
- Disponibilité mise à jour automatiquement lors des emprunts et des retours.

4. Stratégie de Tests

Objectifs des tests :

- Vérifier le bon fonctionnement des fonctionnalités.
- Tester les contraintes métiers.
- Garantir la robustesse des vues et des modèles.

Types de tests réalisés :

- **Tests des modèles :**
- Validation de la création d'objets (Media, Emprunteur, Emprunt).
- Vérification des contraintes métiers : limitation d'emprunts, blocage des emprunteurs.
- **Tests des vues :**
- Vérification du rendu des vues (statuts HTTP 200).
- Tests des formulaires (ajout de membres, médias et emprunts).
- Tests des restrictions (emprunteurs bloqués, médias non disponibles).

Exemple de résultats :

- Les emprunteurs bloqués ou ayant atteint la limite d'emprunts ne peuvent pas emprunter.
- Les jeux de plateau sont exclus des emprunts.
- Les emprunts expirés déclenchent le blocage automatique des membres.

5. Conclusion

Ce projet a permis de moderniser le système de gestion de la médiathèque en le transformant en une application web fonctionnelle, respectant les bonnes pratiques Django et garantissant une meilleure gestion des emprunts et des médias.