# Rapport de travail du devoir « Créer un logiciel de gestion de médiathèque »

## 1/ Étude et correctif du code fourni

Pour ce devoir quelques lignes de code ont été écrites en amont par un autre développeur. Son code n'est pas de la meilleure des qualités mais nous allons reprendre ce dernier pour identifier ce qui peut être gardé et ce qui doit être refait.

Seules les « class models » ont été conservés ici, le système à néanmoins été modifié avec l'ajout d'une classe parente Item qui englobe désormais les classes Livre, Dvd, Cd et Tabletop. Elle nous permet par exemple de regrouper les attributs « name », « disponible » ('availability' dans notre code) ainsi qu'une catégorie correspondante aux différentes classes enfants et une date de parution que nous avons ajouté. On remarque également que le nom des classes enfants est différent, les noms commencent désormais par des majuscules comme souhaité. Les classes ont été correctement réindentées également et la méthode « def__str__ ») a été implantée pour retourner de façon lisible les objets. Pour les jeux de plateau l'attribut availability a été configuré sur None pour qu'ils ne soient pas disponible au prêt comme prévu. Les attributs de date d'emprunt ont été retirés de ces models, ils reviendront dans un nouveau model plus loin. Des contraintes ont été ajoutées pour empêcher la création de deux objets ayant la même catégorie et le même nom.

Le model Member a été réindenté et l'attribut 'name' dupliqué pour pouvoir entrer un nom et un prénom. Ici aussi une contrainte est créée pour éviter à deux membres d'avoir le même nom et prénom.

Et enfin un dernier model Loan à été ajouté, il va nous permettre de lister les prêts avec trois attributs, une clé étrangère qui correspondra à l'item prêté, une deuxième clé correspondante au membre emprunteur et enfin la date de prêt que nous avions supprimé tout à l'heure.

## 2/ Mise en place des fonctionnalités demandées

- Deux applications à déployées

Nous avons bien créé deux applications. 'viewer' qui permettra aux visiteurs de la médiathèque d'avoir un aperçu des objets disponibles actuellement. 'gestion' qui elle permettra l'accès vers la gestion de la base de données, l'accès aux pages de l'application 'gestion' sont protégés à l'aide d'un mot de passe.

- Créer membre-emprunteur / Afficher la liste des membres / Mettre à jour un membre

Via l'application gestion et la mise en place d'un CRUD pour le model Membre tout ceci est possible. Les views sont 'add_memb', 'memb_list', 'edit_memb', 'del_memb'.

- Afficher la liste des médias / Ajouter des médias

Comme pour le Model Member un CRUD à été créé pour Item et ses classes enfants, 'add_item', 'item_lists', 'edit_item', 'del_item'.

- Créer un emprunt pour un média disponible / Rentrer un emprunt

Grace à un if else dans le template de la liste des items qui modifie l'aspect d'un bouton et la direction d'un lien selon la disponibilité d'un item nous accédons soit à la view 'lend_item' qui nous permet d'enregistrez un prêt si l'item est disponible ou nous renvois vers 'return_item' qui nous permettra d'enregistrez le retour d'un item.

- Restrictions sur le nombre de prêt et leur durée

Afin de gérez ces restrictions nous avons intégré un middleware. L'intérêt du middleware est de checker notre restriction à chaque fois que quelqu'un interagira avec notre projet, ce qui permet un rafraichissement en quasi-temps réel tant que l'on utilise la librairie en ligne. Ce middleware est donc chargé de faire tourner la fonction 'check_status_member' qui interroge la base de données, pour chaque membre elle va regarder si un prêt existe, si aucun prêt n'existe le membre n'est pas/plus bloqué. Si un ou des prêts existent pour ce membre on vérifie qu'il n'y en a pas trois ou plus, si c'est égal ou supérieur le membre est bloqué, sinon la fonction continue et vérifie que pour chaque prêt de ce membre la date d'emprunt ne soit pas supérieure à 7 jours, si un ou plusieurs sont supérieurs le membre sera bloqué, sinon comme la fonction se sera assuré que les deux restrictions sont conformes, le membre ne sera pas bloqué.

- Les jeux de plateau ne sont pas concernés par les emprunts

L'attribut 'availability' du model Tabletop à été définit sur None pour empêcher l'utilisation de cet attribut pour cette classe, en plus de cela le bouton permettant le prêt d'un item n'est pas disponible pour les jeux de plateau empêchant l'utilisation de cette fonctionnalité via l'application et du côté visiteur un message a été ajouté pour le public le rappelant lorsque l'on regarde les jeux de plateau disponible.

## 3/ Stratégie de test

Les test mis en place me permettent de vérifier si mes contraintes réagissent bien comme souhaité en faisant remonté des exceptions cela permet par la même occasion de tester la création et la modification d'objets existants. La fonction du middleware est également testée pour vérifier son comportement dans différent cas de figure pour voir si le résultat est celui attendu.

## 4/

La base de données est à l'intérieur du projet. Après avoir importé le projet via Github faites un '$py manage.py migrate' puis un '$py manage.py loaddata db,json' pour entrer la base de données. Puis '$ py manage.py runserver' pour démarrer l'application, côté gestion des identifiants seront demandés avec admin en nom d'utilisateur et mediapass en mot de passe. Ces informations sont disponibles sur le fichier README.md du projet.