جامعة محمد الأول بوجدة
UNIVERSITE MOHAMMED PREMIER OUJDA

Rapport de Projet

Système de Gestion de Bibliothèque

École Nationale des Sciences Appliquées d'Oujda

Première Année Cycle Ingénieur
Filière : Génie Informatique

Réalisé par :
RHIATE Ayoub

Date : 28 juin 2025
Année universitaire : 2024 - 2025

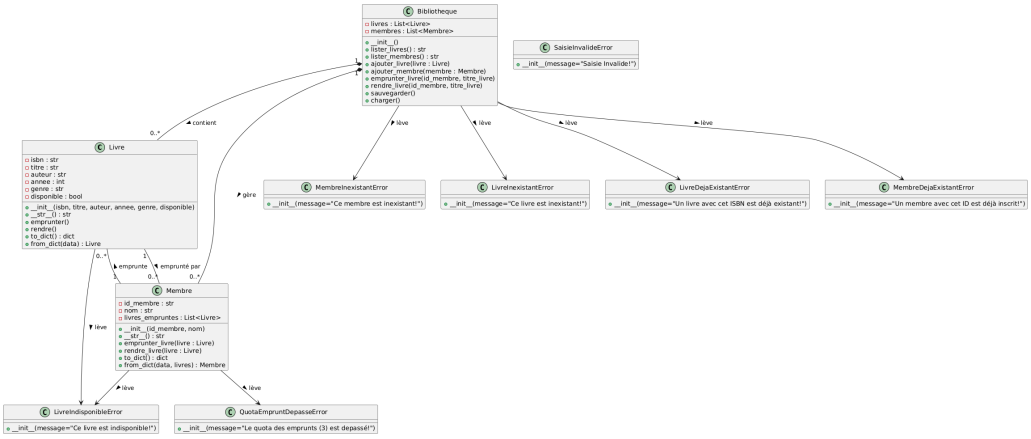
Table des matières

1 Introduction

Ce projet consiste en la réalisation d'une application en ligne de commande (CLI) pour la gestion d'une biblioth'eque. L'application permet d'ajouter des livres, d'inscrire des membres, de gérer les emprunts et les retours, et de visualiser certaines statistiques sous forme graphique.

L'application est d'eveloppée en Python 3 et utilise notamment les bibliothèques `matplotlib`, `csv`, `json`, et `collections`.

2 Diagramme de classes UML



Ce diagramme illustre les relations entre les principales classes : `Livre`, `Membre`, `Bibliotheque` ainsi que les exceptions personnalisées définies dans le fichier `exceptions.py`.

3 Explication des algorithmes clés

3.1 Gestion des emprunts

Lorsqu'un membre emprunte un livre, le programme vérifie d'abord que le livre est disponible, que le membre existe et qu'il n'a pas dépassé le quota d'emprunt. Ensuite, les données sont enregistrées dans un fichier JSON ainsi que dans un fichier CSV pour l'historique.

```
# Methode de la classe Membre
def emprunter_livre(self, livre: Livre):
    if len(self.livres_empruntes) >= 3:
        raise QuotaEmpruntDepasseError()
    if livre.disponible:
        livre.emprunter()
        self.livres_empruntes.append(livre)
    else:
        raise LivreIndisponibleError()

# Methode de la classe Bibliotheque
def emprunter_livre(self, id_membre, titre_livre):
    mbr = None
    lvr = None

    # Recuperer le membre correspondat au ID saisi
    for membre in self.membres:
        if membre.id_membre == id_membre:
```

```
            mbr = membre
            break
        if mbr is None:
            raise MembreInexistantError()

    # Recuperer le livre correspondant au titre saisi
    for livre in self.livres:
        if livre.titre == titre_livre:
            lvr = livre
            break
        if lvr is None:
            raise LivreInexistantError()
    mbr.emprunter_livre(lvr)
    enregistrer_action(mbr.id_membre, lvr.isbn, "emprunt")
```

Code 1 – Méthode de gestion des emprunts

3.2 Gestion des retours

Lors d'un retour, le livre est rendu disponible. La liste des livres empruntés du membre est mise à jour et l'action est également enregistrée dans le fichier CSV.

```
def rendre_livre(self, id_membre, titre_livre):
    mbr = None
    lvr = None

    # Recuperer le membre correspondat au id saisi
    for membre in self.membres:
        if membre.id_membre == id_membre:
        mbr = membre
        break
    if mbr is None:
        raise MembreInexistantError()

    # Recuperer le livre correspondant au titre saisi
    for livre in mbr.livres_empruntes:
        if livre.titre == titre_livre:
        lvr = livre
        break
    if lvr is None:
        raise LivreInexistantError("Ce membre n'a pas emprunte ce livre
            !")
    mbr.rendre_livre(lvr)
    enregistrer_action(mbr.id_membre, lvr.isbn, "retour")
```

Code 2 – Méthode de gestion des retours

3.3 Sauvegarde des données

Pour garantir la persistance des informations d'une session à l'autre, l'application propose une fonctionnalité de sauvegarde automatique à la fermeture. Celle-ci enregistre les listes de livres et de membres dans des fichiers `JSON`. Chaque livre et chaque membre sont convertis en dictionnaires à l'aide de la méthode `to_dict()`, puis exportés en format `.json`.

```
def sauvegarder(self):
    with open("data/livres.json", "w", encoding="utf-8") as f:
        json.dump([livre.to_dict() for livre in self.livres], f, indent
            =4)

    with open("data/membres.json", "w", encoding="utf-8") as f:
        json.dump([membre.to_dict() for membre in self.membres], f,
            indent=4)
```

Code 3 – Méthode de sauvegarde des livres et membres

3.4 Chargement des données

Au lancement de l'application, les fichiers de sauvegarde sont relus afin de restaurer les données précédemment enregistrées. Si les fichiers `livres.json` ou `membres.json` n'existent pas encore (par exemple lors du premier lancement), des listes vides sont initialisées.

```
def charger(self):
    try:
        with open("data/livres.json", "r", encoding="utf-8") as f:
            self.livres = [Livre.from_dict(d) for d in json.load(f)]
    except FileNotFoundError:
        self.livres = []

    try:
        with open("data/membres.json", "r", encoding="utf-8") as f:
            self.membres = [Membre.from_dict(d) for d in json.load(f)]
    except FileNotFoundError:
        self.membres = []
```

Code 4 – Méthode de chargement des livres et membres

Ces deux méthodes constituent le socle de la mémoire persistante du système. Elles permettent une transition fluide entre différentes exécutions du programme et garantissent l'intégrité des données.

3.5 Visualisation des statistiques

Les données enregistrées sont exploitées via la bibliothèque `matplotlib` pour générer des graphiques :

— Diagramme circulaire des livres par genre
— Histogramme des auteurs les plus populaires
— Courbe de l'activité des emprunts sur les 30 derniers jours

4 Captures d'écran des visualisations

4.1 Diagramme des genres

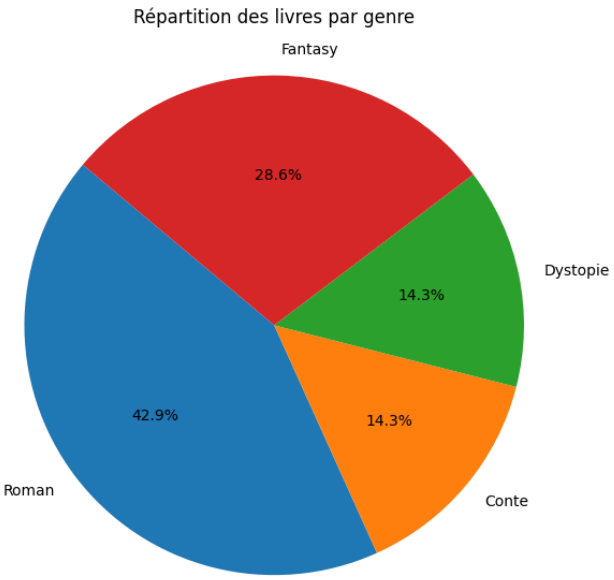


FIGURE 1 – Répartition des livres par genre.

4.2 Top des auteurs

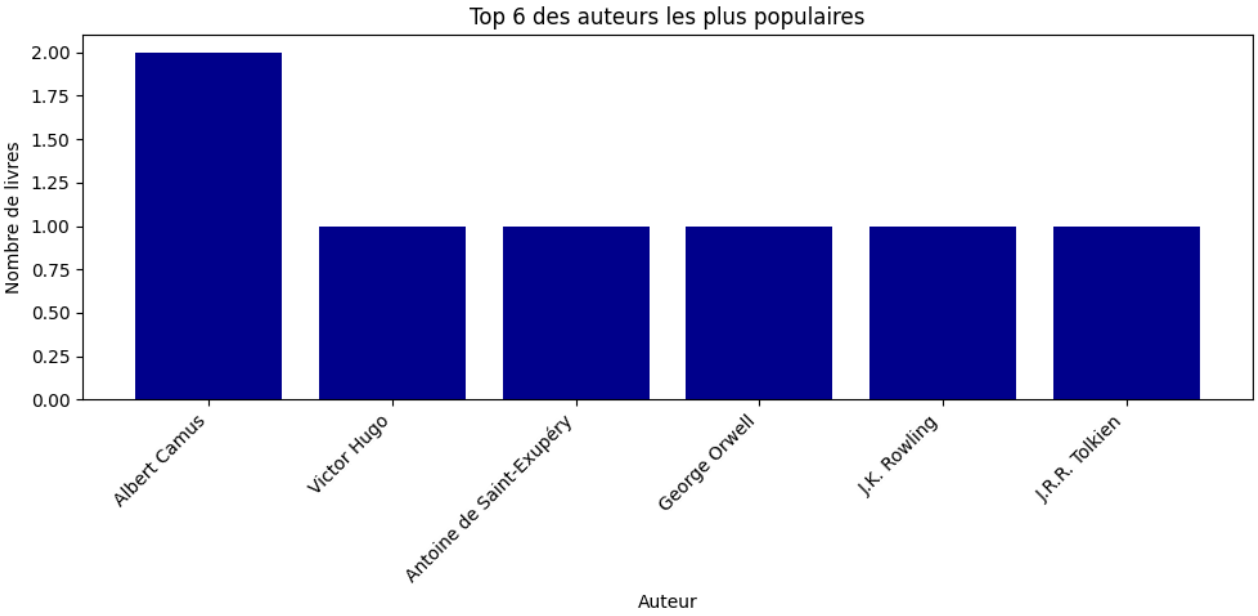


FIGURE 2 – Top 6 des auteurs les plus populaires.

4.3 Activité des emprunts

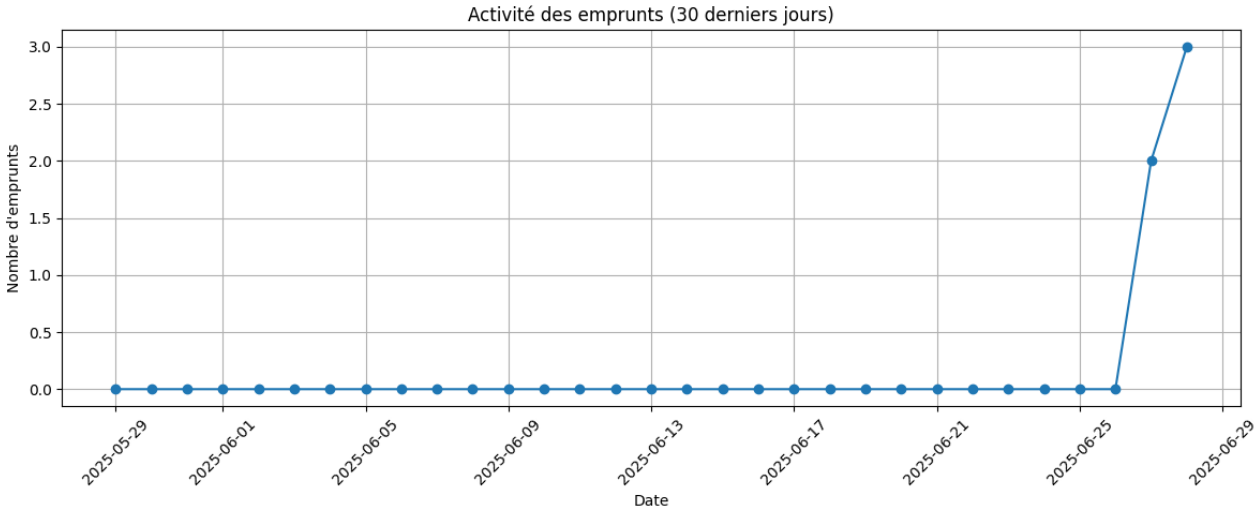


FIGURE 3 – Activité des emprunts sur les 30 derniers jours.

5 Difficultés rencontrées et solutions

Durant le développement, plusieurs défis majeurs ont été relevés et traités efficacement :

— **Sérialisation des objets** : La conversion entre fichiers JSON et objets Python, notamment pour gérer les listes imbriquées comme les livres empruntés, a nécessité une méthode précise. Les fonctions `to_dict()` et `from_dict()` ont été essentielles pour assurer cette sérialisation et désérialisation.

— **Gestion du statut des livres** : Lors du retour d'un livre, il fallait éviter les doublons dans les fichiers JSON. La logique a été renforcée autour des méthodes `emprunter_livre()` et `rendre_livre()` pour mettre à jour correctement la disponibilité sans créer d'entrées redondantes.

— **Validation des entrées utilisateur** : Pour garantir des données propres et cohérentes, plusieurs fonctions de validation ont été mises en place, telles que `saisir_non_vide()`, `valider_annee()`, `valider_id()` et `valider_isbn()`, permettant de contrôler la validité des données saisies en entrée.

Conclusion

Ce projet m'a permis d'approfondir mes compétences en programmation orientée objet, en manipulation de fichiers, et en visualisation de données. Il constitue une base solide pour une future extension vers une application graphique ou web.