UNIVERSITÉ MOHAMMED PREMIER
ÉCOLE NATIONALE DES SCIENCES
APPLIQUÉES - OUJDA
FILIÈRE : GÉNIE INFORMATIQUE

MINI-PROJET :

SYSTÈME DE GESTION DE BIBLIOTHÈQUE

RAPPORT DE PROJET

ANNÉE UNIVERSITAIRE :
2024 / 2025

ENCADRÉ PAR : M. ZAKARIA HAJA

RÉALISÉ PAR : MERIEM SADDOUKI

sommaire

du rapport

01 INTRODUCTION :

Ce mini-projet a été réalisé dans le cadre du module de **Programmation Avancée en Python,** dispensé à l'École Nationale des Sciences Appliquées d'Oujda (**ENSAO**), au sein de la filière Génie Informatique.

Il vise à mettre en pratique les concepts clés de la programmation orientée objet, la manipulation de fichiers, la gestion des erreurs, la visualisation de données, ainsi que la conception d'interfaces utilisateur.

L'environnement Python, grâce à sa richesse en bibliothèques, a permis de construire une application fonctionnelle, interactive et évolutive, répondant aux besoins d'un système de gestion de bibliothèque.

02 Objectif du Projet :

L'objectif principal de ce projet est de développer une application complète de gestion de bibliothèque. Elle permet aux utilisateurs d'ajouter, modifier, emprunter ou restituer des livres, tout en assurant un suivi rigoureux des membres et de l'historique des opérations.

Le projet repose sur plusieurs fondements techniques :

- **POO** (Programmation Orientée Objet) : pour modéliser les entités (livres, membres, bibliothèque).
- **Persistance des données** : via des fichiers TXT, JSON et CSV assurant le stockage et le chargement des données.
- **Gestion des erreurs** : avec des exceptions personnalisées permettant un contrôle fin des anomalies.
- **Visualisation avec Matplotlib** : pour générer des graphiques dynamiques et pertinents.
- **Interface graphique avec Tkinter** : pour offrir une interaction intuitive grâce à des formulaires, boutons, tableaux et messages d'alerte.

03 les diagrammes UML:

diagramme de classe :

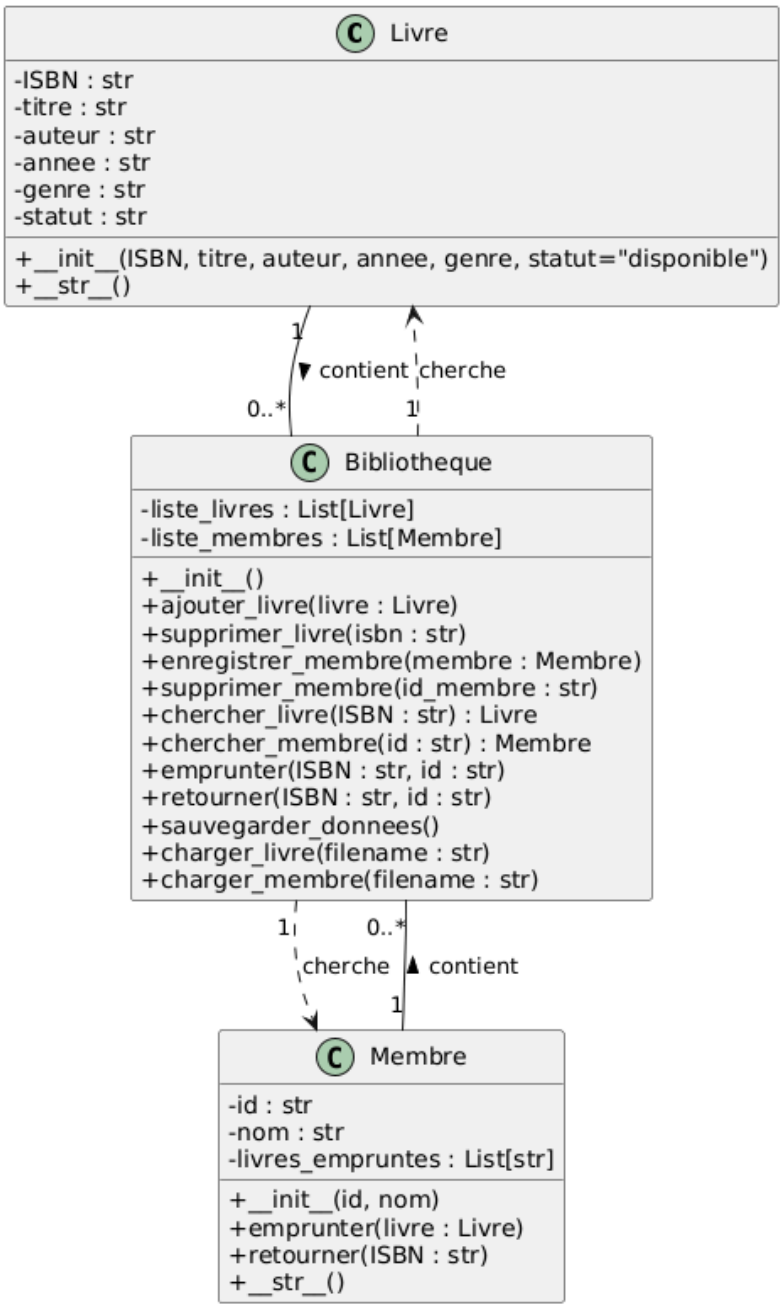
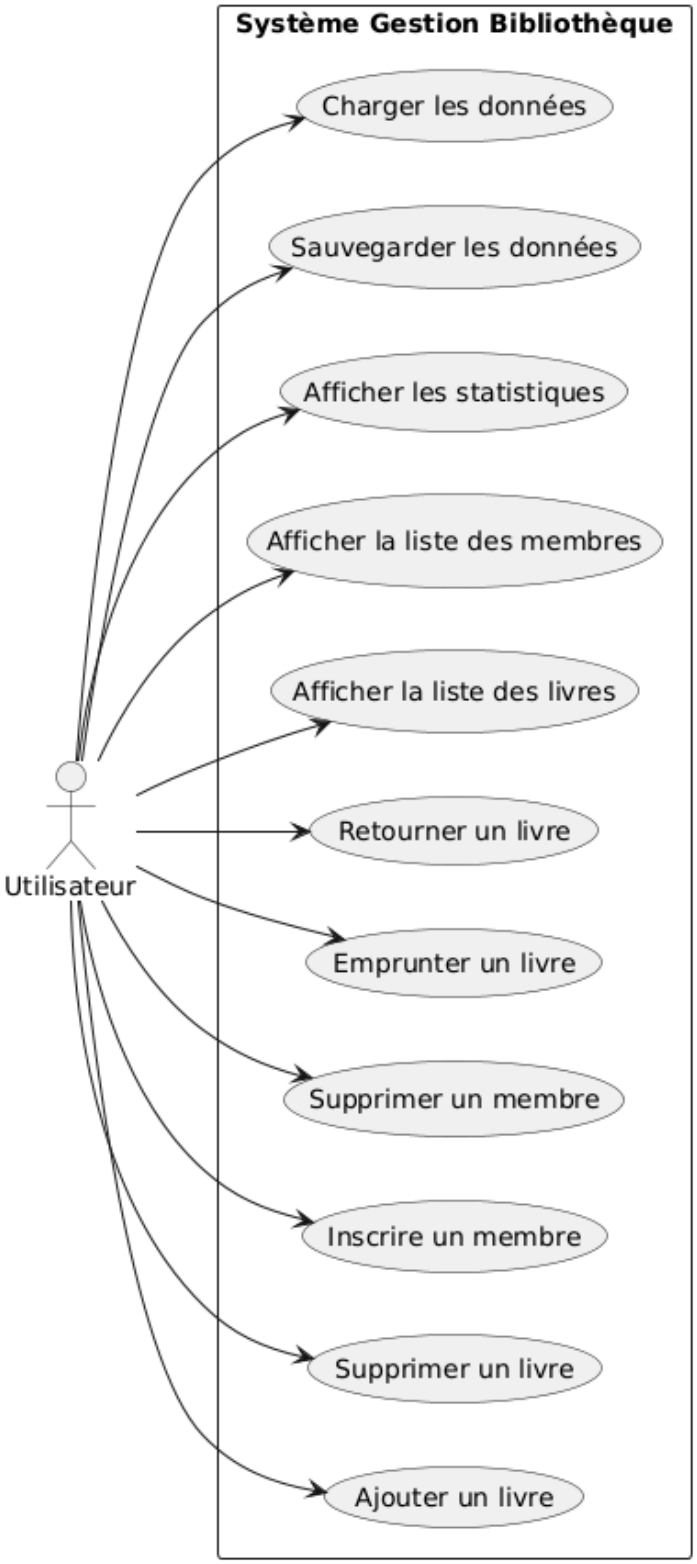
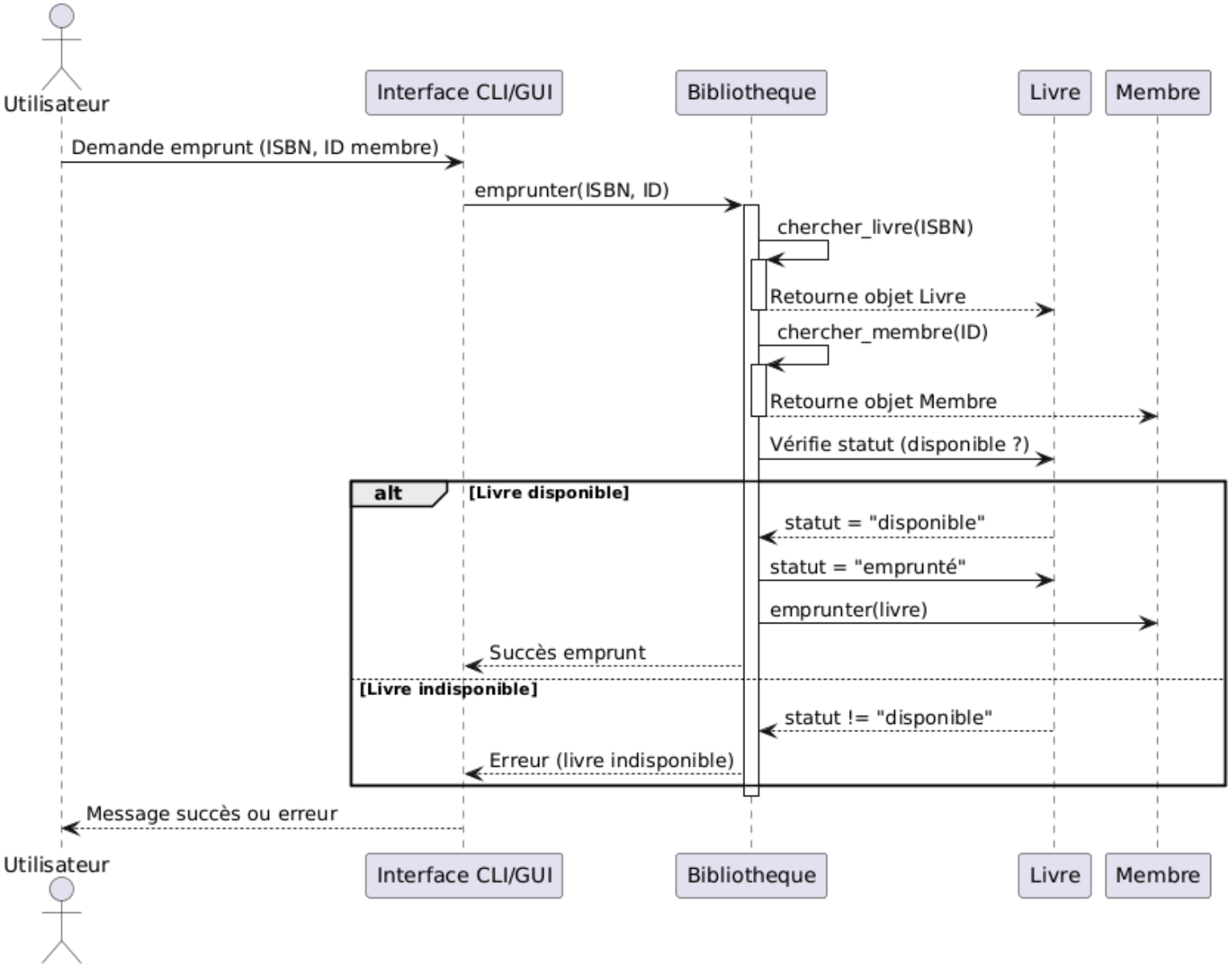


diagramme de cas d'utilisateur :



03 les diagrammes UML:

diagramme de séquence :



04 Explications des algorithmes clés

1. Recherche d'un livre ou d'un membre (chercher_livre/ chercher_membre) :

- **But :**
  Trouver un livre (ou un membre) dans les listes en fonction de son identifiant (ISBN pour livre, ID pour membre).
- **Principe :**
  Parcours linéaire de la liste à la recherche d'un élément avec l'ID correspondant.
- **Détail :**
  On itère sur la liste des livres (ou membres). Si on trouve un livre/membre avec l'identifiant demandé, on le retourne. Sinon, on lève une exception personnalisée (LivreInexistantError ou MembreInexistantError).
- **Complexité :** O(n), avec n le nombre de livres ou membres.

2. Emprunt d'un livre (emprunter) :

- **But :**
  Permettre à un membre d'emprunter un livre s'il est disponible et si le membre n'a pas dépassé son quota.
- **Étapes principales :**
  - Recherche du livre par ISBN.
  - Recherche du membre par ID.
  - Vérification que le livre est disponible (statut == "disponible").
  - Vérification que le membre n'a pas emprunté plus de 3 livres.
  - Mise à jour du statut du livre ("emprunté").
  - Ajout du livre à la liste des livres empruntés du membre.
- **Gestion des erreurs :**
  Utilisation d'exceptions personnalisées pour gérer les cas où le livre est indisponible, le membre inexistant, ou le quota dépassé.

3. Sauvegarde et chargement des données (sauvegarder_livres, charger_livre, etc.)

- **But :**
  Garantir la persistance des données entre les sessions.
- **Principe :**
  - À la fermeture ou à la demande, les listes de livres et de membres sont écrites dans des fichiers texte (livres.txt, membres.txt).
  - Au lancement, ces fichiers sont lus pour recharger les données en mémoire.
- **Format :**
  Chaque ligne contient un enregistrement séparé par un point-virgule (;), facilitant le parsing.

- **Gestion des erreurs :**

Si les fichiers sont absents, un message d'avertissement est affiché, mais le programme continue.

4. Visualisation des statistiques avec Matplotlib :

- **But :** Offrir une interface graphique avec des statistiques visuelles sur la bibliothèque.
- **Exemples d'algorithmes :**
  - ***Calcul de la répartition des livres par genre :*** comptage du nombre de livres par catégorie (dictionnaire genres).
  - ***Top 10 des auteurs les plus présents :*** tri des auteurs par nombre de livres.
  - ***Courbe temporelle des emprunts :*** lecture du fichier historique.csv, comptage des emprunts par date sur 30 jours.
- **Visualisation :**
- Création de graphiques (camembert, histogrammes, courbes) avec sauvegarde automatique des images.

5. Gestion des exceptions personnalisées

- **But :**

Gérer proprement les erreurs spécifiques au domaine métier.

- **Principe :**

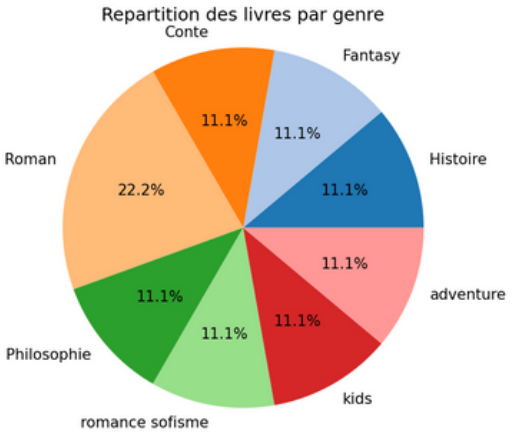
Définition de classes d'exceptions personnalisées (ex : LivreIndisponibleError) pour gérer les cas d'erreurs métiers et simplifier la lecture des erreurs dans l'interface.

- **Avantage :**

Clarifie le flux du programme et facilite la maintenance.

05 Captures d'écran des visualisations :

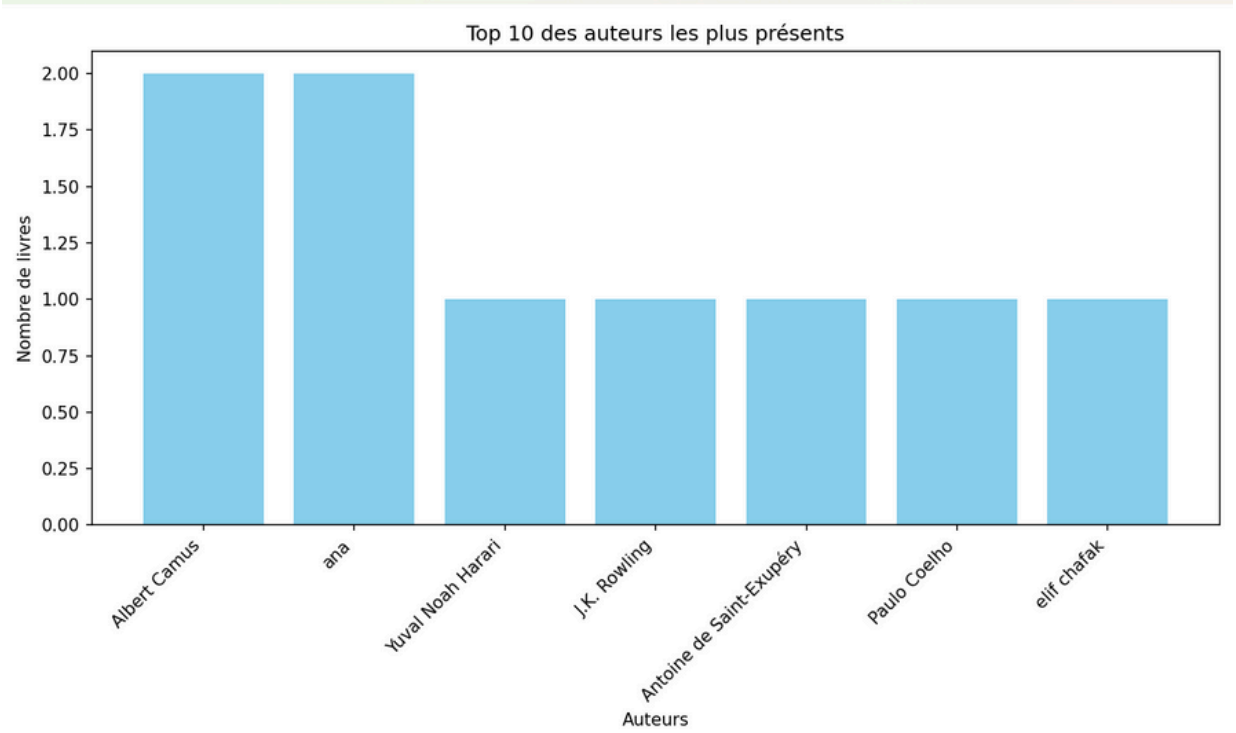
Répartition des livres par genre



Ce graphique circulaire illustre la proportion des livres présents dans chaque genre (roman, science, histoire, etc.). Il permet d'identifier rapidement les types d'ouvrages les plus disponibles.

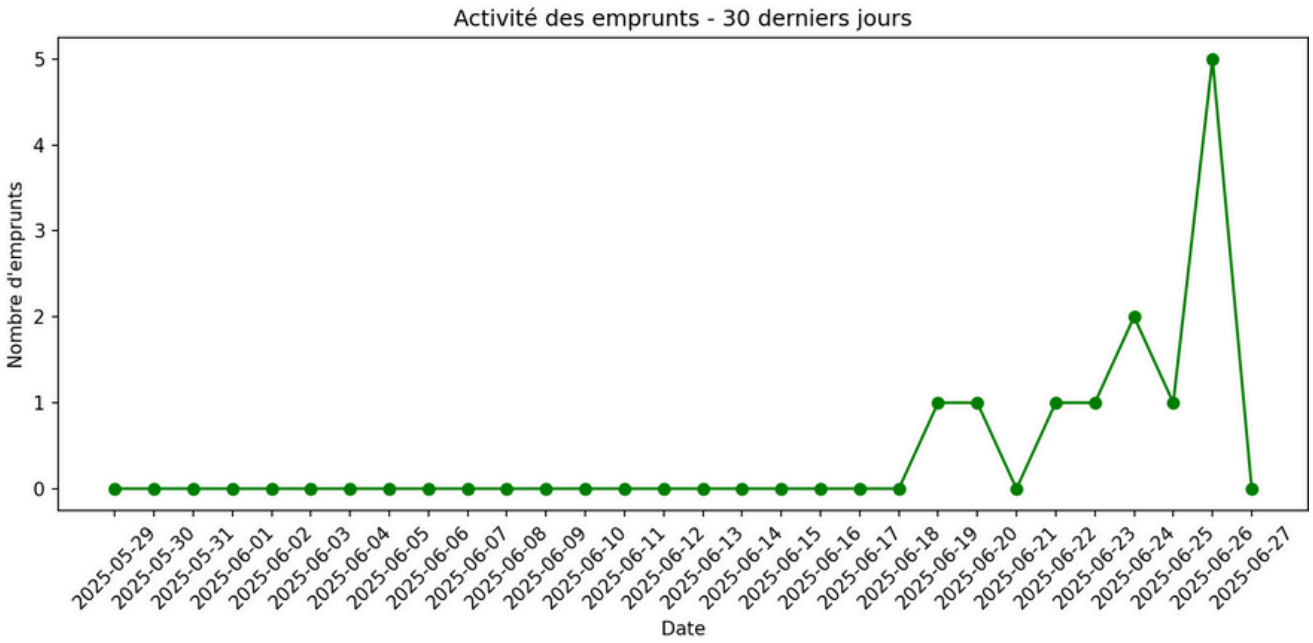
Top 10 des auteurs les plus présents:

Cet histogramme classe les dix auteurs les plus représentés dans la bibliothèque en fonction du nombre de livres disponibles pour chacun. Cela peut orienter les choix de lectures ou d'acquisition.



Activité des emprunts sur les 30 derniers jours

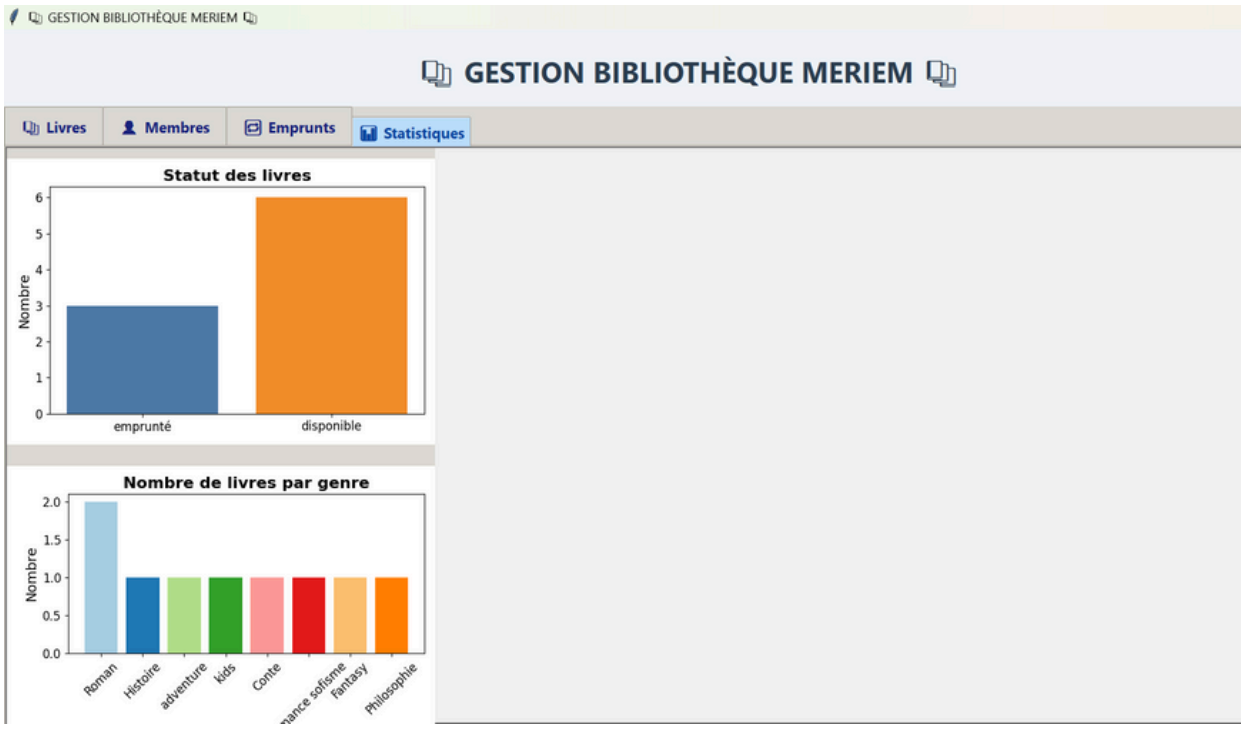
Cette courbe temporelle affiche le nombre d'emprunts enregistrés chaque jour au cours du dernier mois. Elle permet d'observer l'évolution de l'utilisation de la bibliothèque (pics d'activité, creux, etc.).

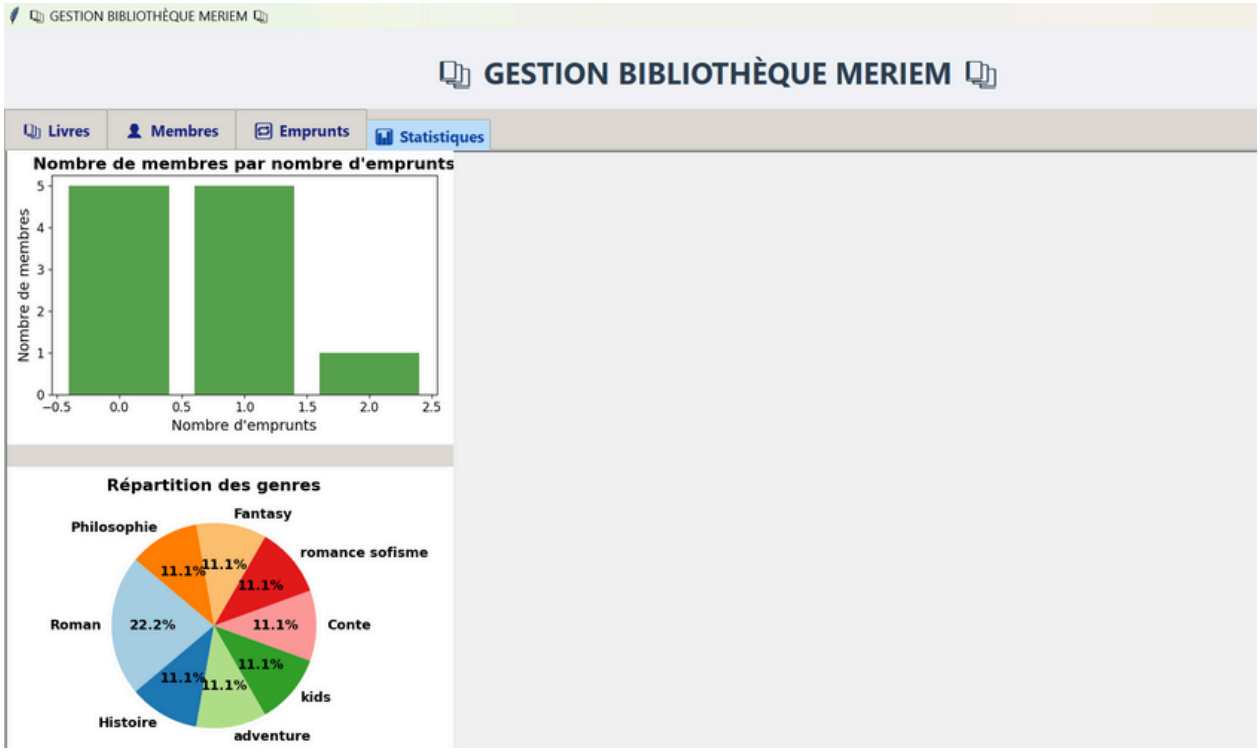


Intégration des visualisations dans Tkinter

L'interface utilisateur propose une section dédiée aux statistiques en temps réel. Elle intègre plusieurs graphiques dans un seul onglet, offrant une vue synthétique sur :

- Le statut des livres (empruntés / disponibles),
- La répartition des livres par genre,
- La distribution des emprunts par membre.





06 Difficultés rencontrées et solutions:

Lors du développement de ce projet de gestion de bibliothèque, plusieurs défis techniques et conceptuels ont été rencontrés. Voici les principales difficultés identifiées, accompagnées des solutions apportées:

1. Sauvegarde et chargement des données

Ce qui m'a posé problème :
Au début, les données ne se rechargeaient pas correctement entre deux exécutions. Par exemple, certains livres n'apparaissaient pas, ou les membres avaient des emprunts « fantômes ». Le souci venait souvent d'un formatage incorrect dans les fichiers .txt.
Ce que j'ai fait :
J'ai pris le temps de bien structurer mes fichiers avec le point-virgule ; comme séparateur unique, et j'ai ajouté des vérifications pour ignorer ou alerter quand une ligne est mal formée. Ça m'a évité beaucoup de bugs silencieux.

2. Courbe des emprunts sur 30 jours

Le bug étrange :
Quand j'ai voulu tracer la courbe des emprunts avec matplotlib, le graphique affichait soit des valeurs vides, soit des dates mal alignées. Je pensais que c'était un problème d'axes, mais en réalité, certaines dates étaient absentes du fichier historique.csv.
Ma solution :
J'ai utilisé Counter avec une liste complète des 30 derniers jours, même ceux sans emprunt (valeur par défaut à 0). Grâce à ça, la courbe est devenue fluide et continue.

3. Problème avec l'utilisation de TKINTER:

Une vraie nouveauté pour moi :
Avant ce projet, je n'avais jamais utilisé Tkinter. J'étais un peu perdue au début avec la structure des frames, les grid, et surtout la gestion des événements (command, bind, etc.). Je ne savais même pas comment faire un bouton qui réagit correctement ou comment actualiser une Treeview.
Ce que j'ai appris :
Petit à petit, j'ai compris comment organiser une interface en onglets, comment styliser les widgets, et comment connecter l'interface à ma logique Python. J'ai aussi découvert pas mal d'astuces visuelles pour que l'application soit agréable à utiliser. Finalement, même si ça m'a demandé du temps, j'ai beaucoup aimé travailler avec Tkinter.

07 Points forts du projet

1. Un système complet, de bout en bout :

L'application couvre tout le cycle de vie d'un livre dans une bibliothèque : **ajout**, **suppression**, **emprunt**, **retour**, et **suivi historique**. Cela m'a permis d'appliquer de manière concrète les concepts appris en **Programmation Orientée Objet.**

2. Gestion persistante des données :

L'utilisation de fichiers **TXT, CSV et JSON** pour sauvegarder les livres, membres et historiques garantit la continuité des données. À chaque relance de l'application, les données sont automatiquement rechargées : pas besoin de tout recommencer.

3. Visualisation claire et pertinente :

Grâce à **Matplotlib**, j'ai pu produire des statistiques visuelles attractives : camembert des genres, histogrammes des auteurs les plus fréquents, courbes d'emprunts. Cela donne une vraie valeur ajoutée au système, surtout pour un gestionnaire.

4. Interface graphique moderne et fonctionnelle :

La réalisation d'une interface graphique avec **Tkinter**, bien stylisée et intuitive, rend l'application accessible même aux utilisateurs non techniques. Le passage entre les onglets Livres, Membres, Emprunts et Statistiques est fluide.

5. Robustesse grâce à la gestion d'exceptions :

L'application gère les erreurs de manière élégante, avec des messages clairs pour l'utilisateur. Les exceptions personnalisées comme **LivreInexistantError** ou **QuotaEmpruntDepasseError** permettent une meilleure fiabilité du code.

08 Conclusion

Ce projet de gestion de bibliothèque a été une véritable opportunité pour mettre en pratique les notions avancées de Python acquises durant le semestre. Il m'a permis de consolider mes compétences en programmation orientée objet, en gestion des fichiers (JSON, CSV, TXT) et en manipulation des structures de données.
L'ajout d'une interface graphique avec Tkinter, bien que nouveau pour moi, m'a permis de découvrir l'importance de l'expérience utilisateur dans une application. L'intégration de visualisations statistiques via Matplotlib a, quant à elle, enrichi le projet en offrant une lecture claire et intuitive des données.
Malgré les difficultés techniques rencontrées (notamment avec Tkinter ou la structuration du code), ce projet a été une expérience d'apprentissage riche et formatrice. Il m'a également permis de mieux comprendre le cycle complet de développement d'une application : de la conception jusqu'à l'interface utilisateur, en passant par la gestion des exceptions et la persistance des données.
En somme, cette réalisation représente une étape importante dans mon parcours, en me préparant à développer des projets plus complexes et orientés utilisateur.