Université Mohammed Premier
École Nationale des Sciences Appliquées d'Oujda
Filière : Génie Informatique – GI3
Année universitaire : 2024 / 2025

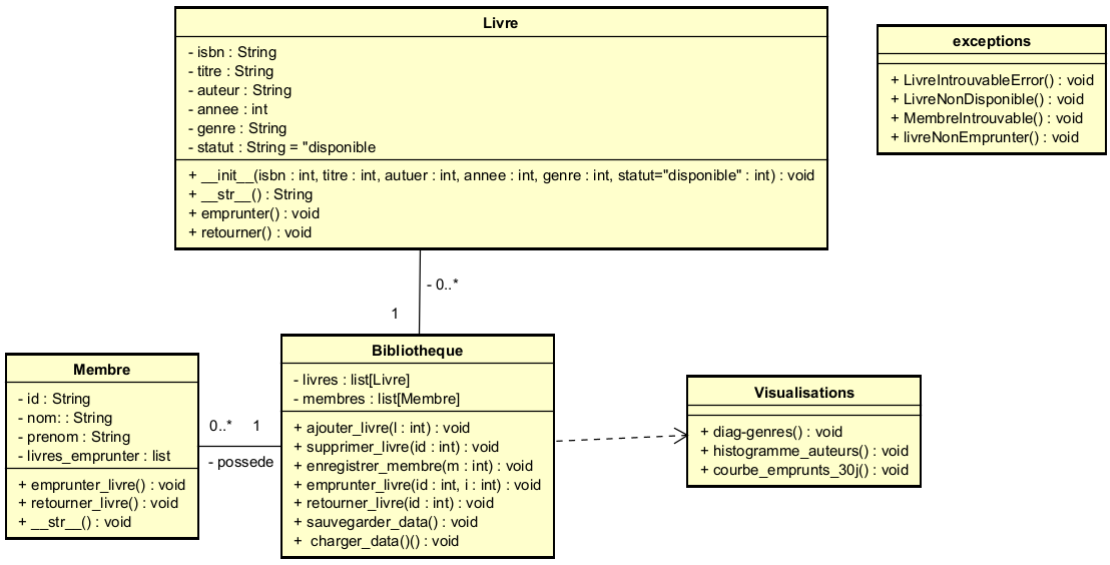
Système de Gestion de Bibliothèque

Projet réalisé dans le cadre du module Programmation Avancée en Python

Réalisé par : **Ouissal Khfifi**
Date de rendu : 28 juin 2025

1. Diagramme de classes UML

Le diagramme ci-dessous représente les principales classes utilisées dans le projet, ainsi que leurs relations d'héritage ou d'association. Ce diagramme a été réalisé avec l'outil astah



Bibliotheque contient des collections de Livre et Membre

Membre possède une liste de livres empruntés (livres_empruntes)

Le statut d'un Livre indique s'il est disponible ou emprunté

2. Explications des algorithmes clés

Voici quelques extraits représentatifs des algorithmes essentiels du projet.

a) Ajout d'un livre

L'algorithme permet d'ajouter un livre à la bibliothèque, tout en vérifiant qu'un livre avec le même ISBN n'existe pas déjà dans les fichiers.

```
def ajouter_livre(self, livre):
    self.livres.append(livre)
```

b) Emprunt d'un livre

```
def emprunter_livre(self,id_membre,isbn):
   membre=next((m for m in self.membres if m.id==id_membre), None)
   if not membre:
      raise MembreIntrouvableError(...)
   livre=next((l for l in self.livres if l.isbn ==isbn), None)
   if not livre:
      raise LivreIntrouvableError(...)
   if not membre.emprunter_livre(livre):
      raise livreNonEmprunterError(...)
   return True
```

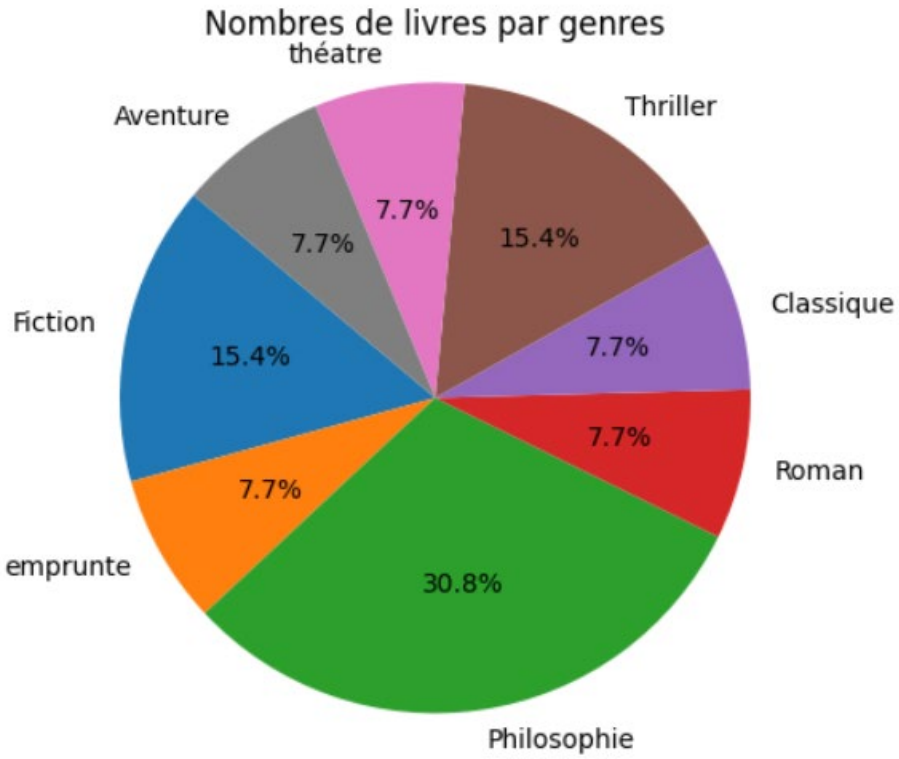
c) Statistiques

```
def diag_genres(ax):
   genres = []
   with open("data/livres.txt") as f:
      for line in f:
         genres.append(line.strip().split(";")[4])
   counts = Counter(genres)
   ax.pie(counts.values(), labels=counts.keys(), autopct="%1.1f%%")
```

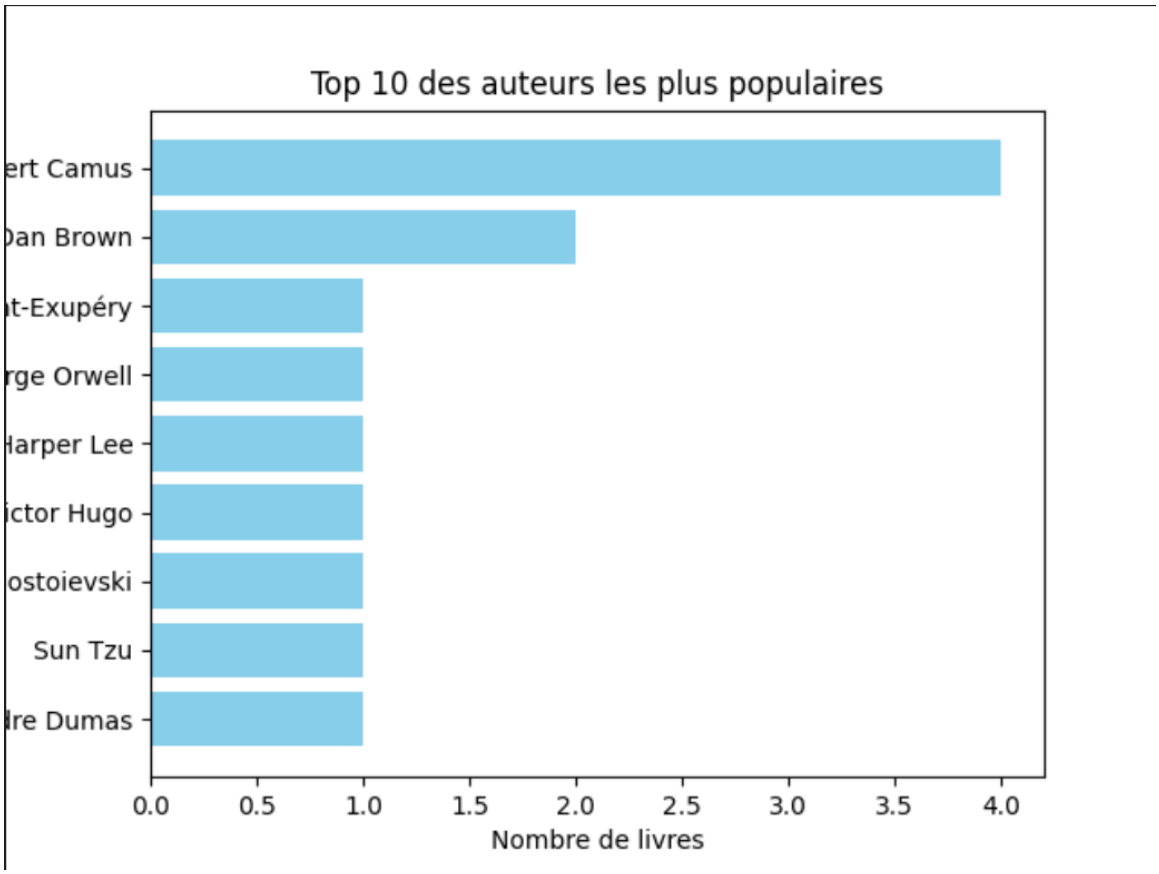
3. Captures d'écran des visualisations

Voici les différentes visualisations générées par l'application :

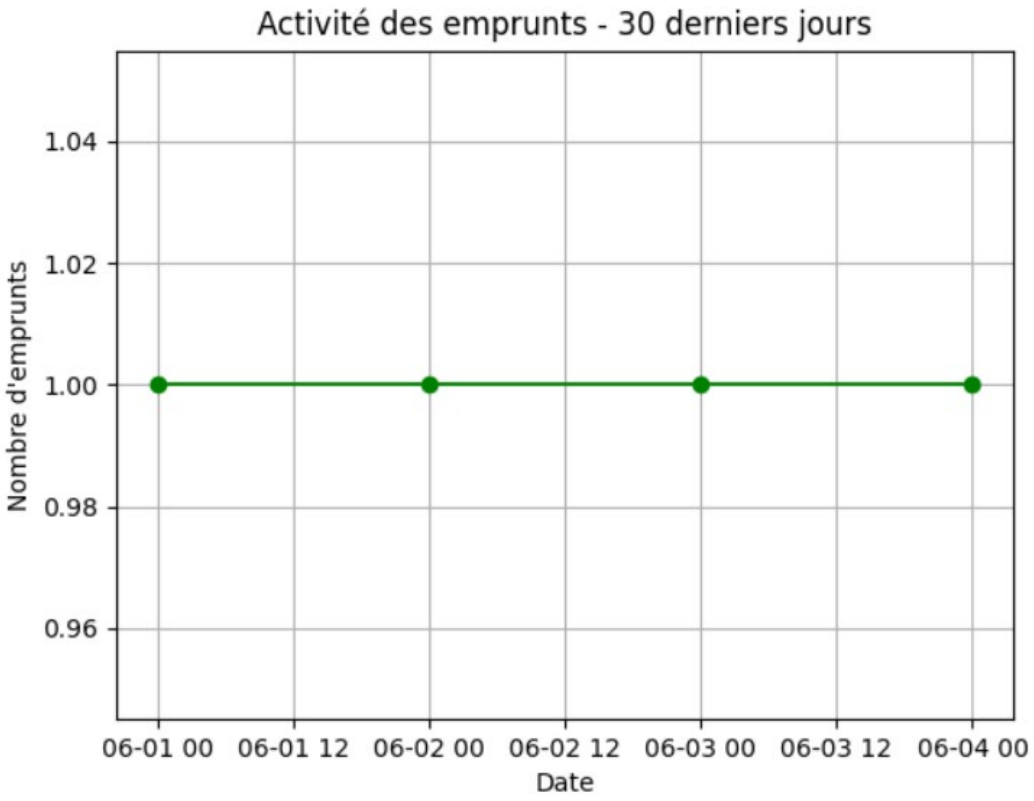
- Diagramme circulaire : répartition des livres par genre



- Histogramme : top 10 des auteurs



- Courbe : activité d'emprunt sur 30 jours



4. Gestion des interactions dans l'interface graphique

L'application utilise **CustomTkinter** pour créer une interface moderne avec une barre latérale contenant plusieurs boutons, chacun lié à une fonction précise.

Fonctionnement des boutons

- Chaque bouton de la sidebar est configuré avec un paramètre command qui déclenche une fonction Python spécifique.
- Ces fonctions réalisent trois actions principales :
    1. **Effacer** le contenu actuel de la zone principale (main_content).
    2. **Afficher** un formulaire ou une vue correspondant à l'action choisie (ex : formulaire ajout de livre, liste des membres).
    3. **Gérer** la saisie utilisateur et la validation (ex : ajout, emprunt, retour) avec retour visuel via un label de message.

Exemple de cycle lors du clic sur « Ajouter un livre »

- L'utilisateur clique sur « Ajouter un livre ».
- La fonction ajouter_livre() appelle afficher_formulaire_ajout_livre() qui construit dynamiquement un formulaire avec les champs ISBN, Titre, Auteur, Année, Genre.
- L'utilisateur remplit les champs et clique sur « Valider ».
- La fonction interne valider() récupère les données, vérifie qu'elles ne sont pas vides, crée une instance Livre, l'ajoute à la bibliothèque via biblio.ajouter_livre(livre).
- La bibliothèque est sauvegardée dans un fichier texte via biblio.sauvegarder_data().
- Un message de confirmation s'affiche.

Gestion des erreurs et retours utilisateur

- En cas de champ vide ou erreur, un message d'erreur rouge apparaît sous le formulaire.
- Lors d'une opération réussie, un message vert confirme l'action.
- Les exceptions (ex : emprunt d'un livre déjà emprunté) sont capturées et affichées à l'utilisateur.

5. Difficultés rencontrées et solutions

a) Interface dynamique avec CustomTkinter

Problème : afficher plusieurs formulaires dans une même fenêtre.
Solution : utilisation d'un frame principal avec .pack_forget() et .pack().

b) Actualisation des statistiques

Problème : les graphiques n'étaient pas mis à jour après une action.
Solution : appel explicite des fonctions de visualisation après chaque emprunt ou retour.

c) Gestion des fichiers

Problème : risque de doublons dans les fichiers textes.
Solution : vérification de l'unicité des identifiants (ISBN, ID) avant insertion.

5. Conclusion

Ce projet m'a permis de mettre en œuvre la programmation orientée objet, la création d'interfaces graphiques avec CustomTkinter, et l'utilisation de bibliothèques comme matplotlib pour générer des statistiques. Parmi les améliorations envisageables : intégrer une base de données SQLite, ajouter une authentification, ou exporter les données en PDF.