جامعة محمد الأول بوجدة
UNIVERSITE MOHAMMED PREMIER OUJDA
المدرسة الوطنية للعلوم التطبيقية
École Nationale des Sciences Appliquées d'Oujda

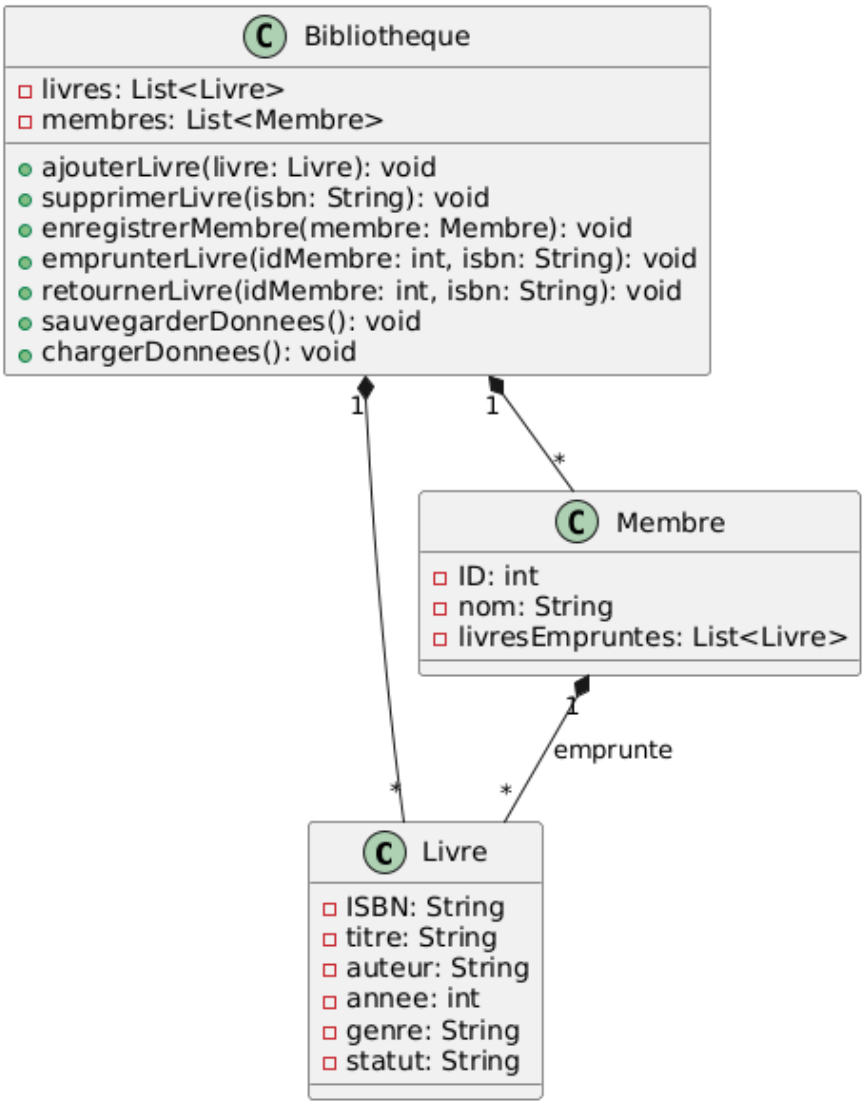
المدرسة الوطنية للتجارة والتسيير
École Nationale de Commerce et de Gestion

Mini projet d'application Système de Gestion de Bibliothèque

Realisé par: Yachou Hajar

Module: Python

Diagramme du classes



Livre

- Attributs : ISBN, titre, auteur, année, genre, statut (disponible ou emprunté).
- Représente un ouvrage présent dans la bibliothèque.

Membre

- Attributs : ID, nom, liste des livres empruntés.
- Un membre peut emprunter au max 3 livres

Bibliotheque

- Attributs : listes de livres et de membres.
- Méthodes :

- ajouter/supprimer livre :pour gérer les ouvrages.
- Enregister membre : pour ajouter des membres.
- Emprunter/retourner livre :pour gérer les prêts.
- Sauvegarder/Charger : pour persister les données des livres, membres et les emprunts

Explications des algorithmes clés

✅ 1. Algorithme d'emprunt de livre

Fichier concerné : `emprunts_tab` + méthode `Bibliotheque.emprunt()`

Rôle :

- Vérifie si le livre est disponible (`statut == 0`)
- Vérifie si le membre existe et n'a pas dépassé le quota
- Change le statut du livre en `1` (= emprunté)
- Ajoute le livre à la liste `list_emprunte` du membre

Pourquoi c'est clé :

C'est le cœur du système de gestion, il inclut des vérifications, des exceptions personnalisées, une mise à jour de l'état global et une persistance des données.

✅ 2. Algorithme de retour de livre

Fichier concerné : `emprunts_tab` + méthode `Bibliotheque.retour()`

Rôle :

- Vérifie si le livre est dans la liste des emprunts du membre
- Met à jour son statut à disponible (`0`)
- Sauvegarde les nouvelles données

Pourquoi c'est clé :

Complémentaire à l'emprunt, c'est ce qui permet de libérer un livre et de maintenir la cohérence des données.

✅ 3. Chargement et sauvegarde des données (fichiers `.txt`)

Méthodes :

- `Bibliotheque.charger_livres()`
- `Bibliotheque.sauvegarder_livre()`
- `Bibliotheque.charger_membres()`

- `Bibliotheque.sauvegarder_membre()`

Rôle :

- Lire et écrire les livres et membres depuis/vers des fichiers texte
- Mettre à jour la mémoire de l'application

Pourquoi c'est clé :

Ces fonctions assurent la persistance des données même après fermeture de l'application.

✅ 4. Génération des statistiques (graphique)

Fonctions :

- `genre_pie(bibliotheque)`
- `année_bar(bibliotheque)`

Rôle :

- Parcourir la liste des livres
- Compter et regrouper les genres et années
- Générer et sauvegarder les graphiques avec Matplotlib

Pourquoi c'est clé :

Elles transforment les données en visualisations compréhensibles, ce qui est un plus dans l'interface.

✅ 6. Fonction `refresh_all_tabs()` (Mise à jour dynamique de l'interface)

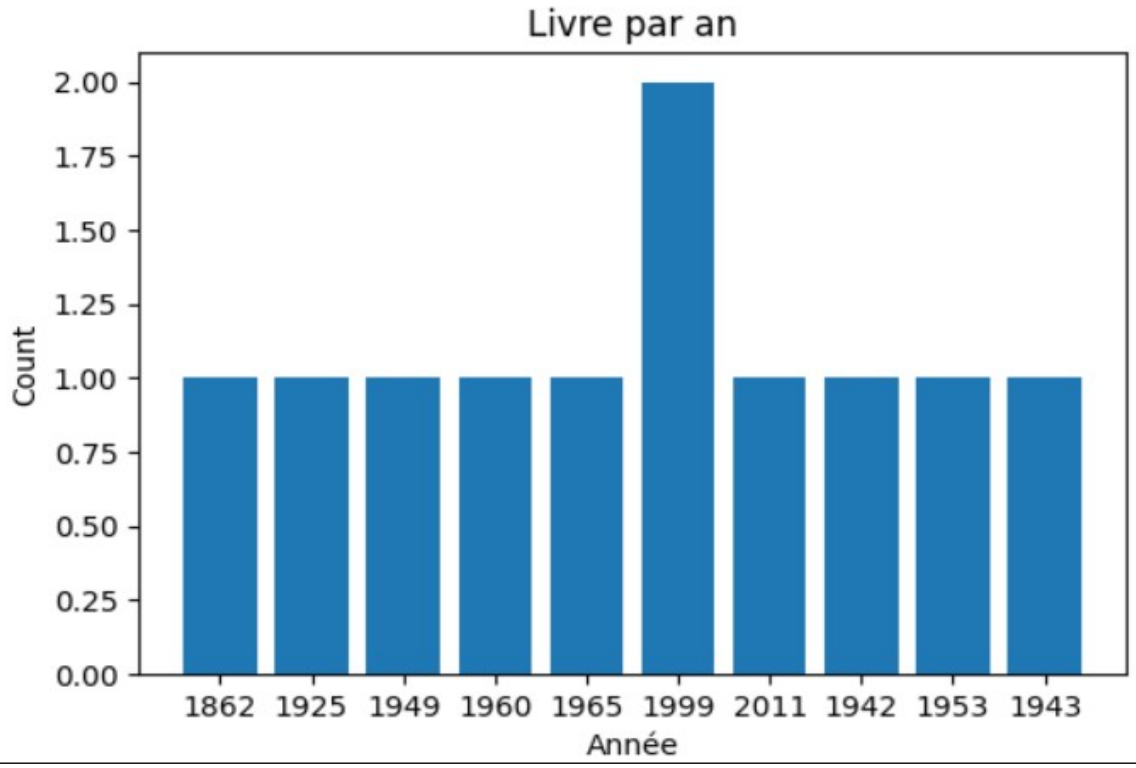
Fichier : `main.py`

Rôle :

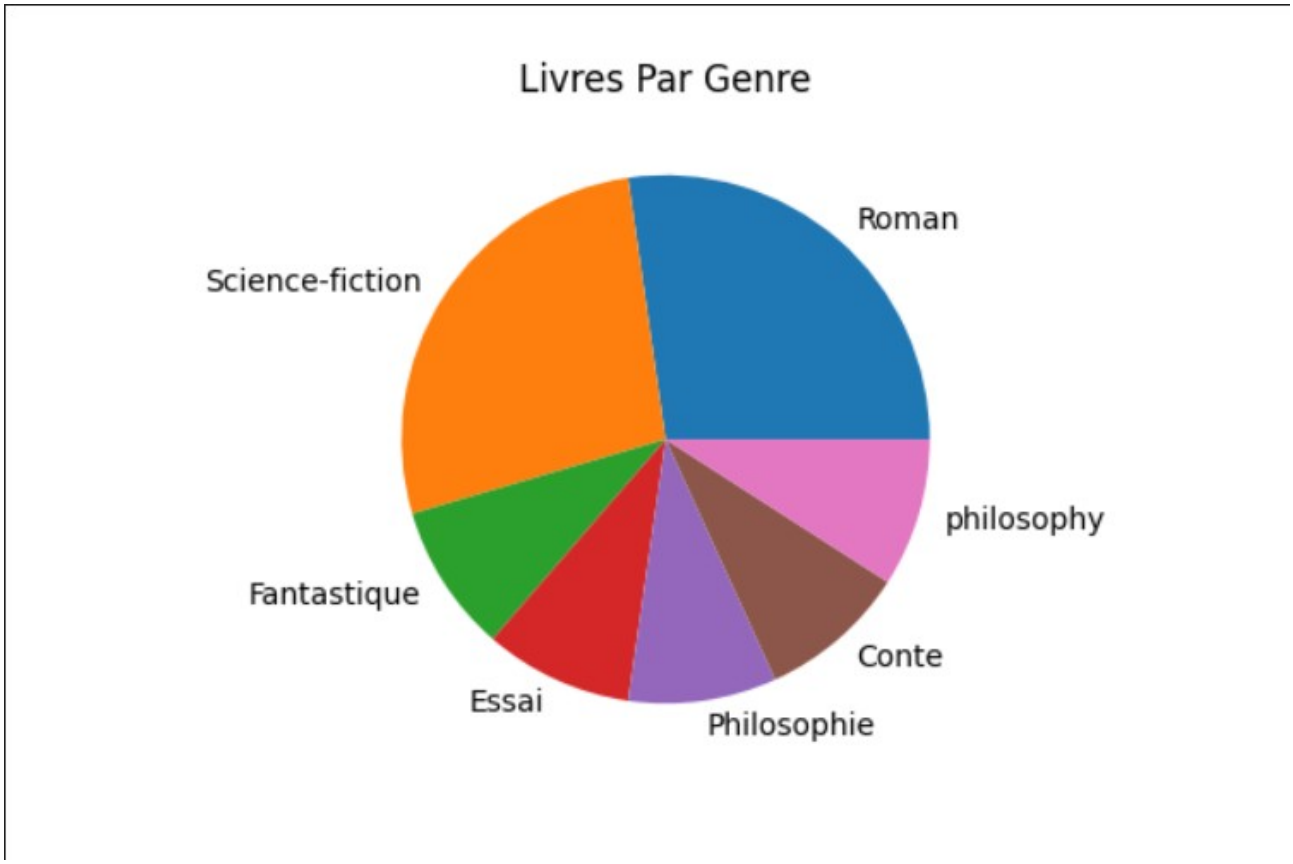
- Recharge tous les onglets (`Accueil`, `Livres`, `Membres`, `Emprunts`, `Statistiques`)
- Vide chaque `tab` de ses composants (`destroy()`), puis les reconstruit avec les données les plus à jour
- Sert de **callback** pour forcer l'actualisation après chaque emprunt/retour

Visualisations

Bar chart : Livre par année.



Pie chart : Livre par genre.



Difficultés rencontrées et solutions

1. Découverte accélérée de la bibliothèque `customtkinter`

L’une des principales difficultés rencontrées lors du développement de l’interface graphique a été la **nécessité d’apprendre et maîtriser `customtkinter` en très peu de temps**. Cette bibliothèque, bien que puissante et moderne, demande une bonne compréhension des concepts fondamentaux de `tkinter`, tels que les frames, les layouts, les callbacks, et la gestion d'événements.

✅ Solution adoptée :

Pour surmonter cette contrainte, j’ai eu recours à une combinaison de **vidéos tutoriels YouTube**, **discussions StackOverflow**, et l’assistance de **ChatGPT** pour comprendre la logique de création de composants, les bonnes pratiques d’organisation d’interface, ainsi que l’intégration fluide avec les données en Python.

2. Mise à jour dynamique de l’interface (`refresh`)

Un autre problème majeur concernait **le rafraîchissement des onglets**. Au départ, après un emprunt ou un retour de livre dans l'onglet "Emprunts", les modifications n’étaient **pas visibles dans les autres onglets** (Livre et Membre tab) tant que l’application n’était pas redémarrée. Cela nuisait fortement à l’expérience utilisateur.

✅ Solution adoptée :

Après plusieurs essais et recherches, la solution retenue a été la **création d’une fonction centrale `refresh_all_tabs()`**. Cette méthode détruit et reconstruit chaque onglet de l’interface avec les données les plus à jour. Grâce à ce mécanisme, l'application garantit une synchronisation complète entre les actions de l’utilisateur et l’interface affichée. Ce correctif a été inspiré et validé à l’aide d’échanges sur StackOverflow, de tests progressifs, et de l’analyse logique de fonctionnement guidée par ChatGPT.