

**Rapport de Projet**

**Application de Gestion de Bibliotheques par Python**

Table des matières

[I. I. Introduction 2](#_Toc201855104)

[II. .Objectifs 2](#_Toc201855106)

[**III.** Fonctionnalités principales 2](#_Toc201855111)

[A. 1. Gestion des livres 2](#_Toc201855112)

[B. 2. Gestion des membres 2](#_Toc201855117)

[C. 3. Gestion des emprunts 2](#_Toc201855123)

[D. 4. Statistiques et visualisations 2](#_Toc201855134)

[IV. Modélisation du système : 3](#_Toc201855140)

[A. Diagrammes UML : 3](#_Toc201855141)

[V. 2. Explication des algorithmes clés 4](#_Toc201855142)

[A. a. Ajout d’un Livre 4](#_Toc201855143)

[B. b. Emprunt et Retour de Livre 4](#_Toc201855144)

[C. c. Affichage des Graphes Statistiques 4](#_Toc201855145)

[D. d. Chargement et Sauvegarde des Données 5](#_Toc201855146)

[VI. Captures d’ecran des visualisations : 5](#_Toc201855147)

[VII. Problemes et les solutions : 7](#_Toc201855148)

Réalisé par : CHTIOUI Maryam

# I. Introduction

### Ce projet présente une application de gestion de bibliothèque simple et interactive, dotée d’une interface graphique (Tkinter) et d’une interface en console pour plus de flexibilité. Elle permet la gestion des livres, des membres, des emprunts/retours, ainsi que l’affichage de statistiques visuelles. L’architecture vise à être claire, évolutive et robuste.

# .Objectifs

### Proposer deux interfaces : graphique et console.

### Gérer efficacement livres, membres et emprunts.

### Offrir un suivi statistique visuel des données.

### Assurer la robustesse via une gestion avancée des erreurs.

# Fonctionnalités principales

## 1. Gestion des livres

### Ajout de livres avec validation des champs (ISBN, titre, auteur, année, genre).

### Affichage des livres dans un tableau clair.

### Chaque livre est identifié de façon unique par son ISBN.

### Possibilité d’ajouter (dans consol+Tkinter)et supprimer des livres dans l’interface Tkinter

## 2. Gestion des membres

### Inscription de nouveaux membres avec vérification des identifiants.

### Gestion des emprunts et retours par titre et ID membre.

### Enregistrement des opérations dans un fichier historique.

### Limitation : un membre peut emprunter jusqu’à 7 livres simultanément.

### Possibilité d’ajouter, modifier et supprimer des membres.

## 3. Gestion des emprunts

### Un livre ne peut être emprunté que s’il est disponible.

### Seuls les membres inscrits peuvent emprunter.

### Lors d’un emprunt :

### Le statut du livre passe à « Emprunté ».

### L’ISBN est ajouté à la liste des emprunts du membre.

### Une entrée est ajoutée au fichier historique (date; ISBN; id\_membre; EMPRUNT).

### Lors d’un retour :

### Le livre est marqué comme « Disponible ».

### L’ISBN est retiré de la liste des emprunts du membre.

### Une entrée est ajoutée au fichier historique (date; ISBN; id\_membre; RETOUR).

## 4. Statistiques et visualisations

# **Objectif :** Offrir une vue analytique claire à l’administrateur.

# **Fonctionnalités :**

# Diagramme circulaire montrant la répartition des livres par genre.

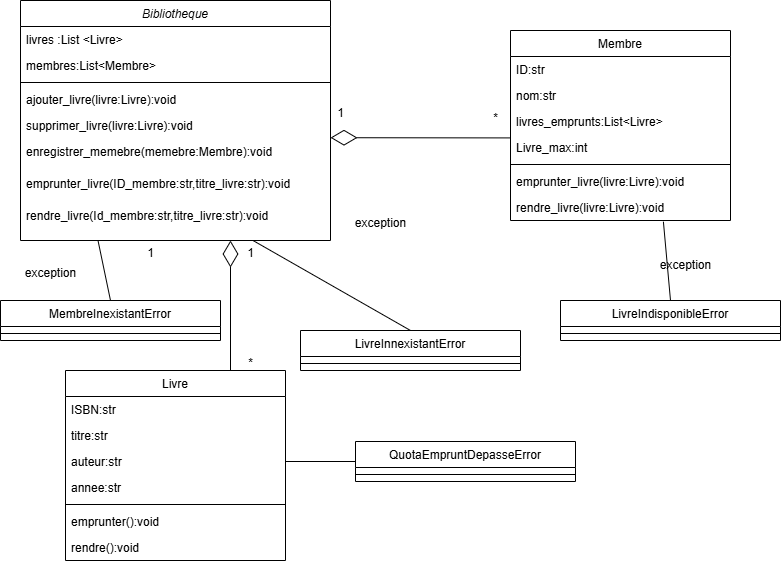
# Histogramme affichant le top 10 des auteurs les plus populaires.

# Courbe représentant le nombre d’emprunts réalisés sur les 30 derniers jours, basée sur les données du fichier historique.

# Modélisation du système :

## Diagrammes UML :

**Diagramme de classe :**

****

# 2. Explication des algorithmes clés

## Ajout d’un Livre

Dans **interface\_graphique.py**, lorsqu’un utilisateur saisit un nouveau livre, on récupère les informations depuis les champs Entry, on les nettoie avec .strip() pour retirer les espaces superflus, puis on instancie un objet Livre.

**infos = [entry.get().strip() for entry in self.entrees\_livre.values()]**

**livre = Livre(\*infos)**

**self.biblio.ajouter\_livre(livre)**

Le livre est ensuite inséré dans la table graphique et sauvegardé dans le fichier texte via la fonction :

**sauvegarder\_livres(self.biblio.livres)**

## Emprunt et Retour de Livre

Dans **interface\_graphique.py**, lorsqu’un livre est emprunté ou rendu, le programme :

1. Identifie le membre à partir de son ID
2. Cherche le livre correspondant au titre saisi
3. Met à jour les états (emprunté ou disponible)
4. Enregistre l’action dans un fichier CSV via ajouter\_historique :

**self.biblio.emprunter\_livre(ID, titre)**

**ajouter\_historique(date, livre.ISBN, ID, "emprunt")**

Dans **bibliotheque.py,** pour éviter les erreurs dues aux majuscules, accents ou espaces, les titres sont nettoyés avec .lower().strip() avant comparaison

## Affichage des Graphes Statistiques

Dans visualisations.py, les graphiques sont créés avec matplotlib, puis **renvoyés sous forme d’objet Figure**. Ensuite, dans interface\_graphique.py, ces figures sont intégrées **directement dans l’interface graphique Tkinter**, sans ouvrir une nouvelle fenêtre :

canvas = FigureCanvasTkAgg(fig, master=self.frame\_graph)

canvas.draw()

canvas.get\_tk\_widget().pack(fill='both', expand=True)

## Chargement et Sauvegarde des Données

Dans **gestion\_donnees.py**, les fonctions comme **charger\_livres()** ou **sauvegarder\_livres()** utilisent **le module csv** pour lire et écrire les fichiers livres.txt, membres.txt, historique.csv

with open("livres.txt", "r", encoding="utf-8") as f:

lignes = f.readlines()

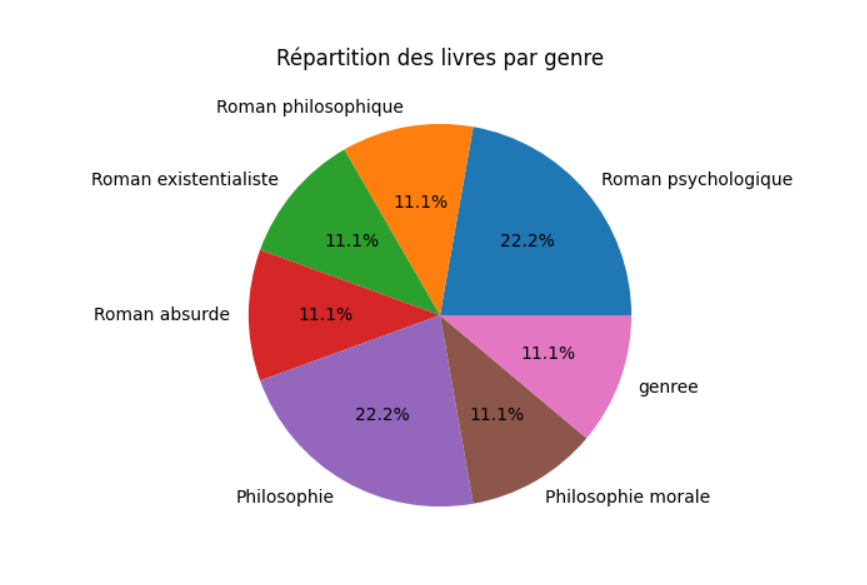
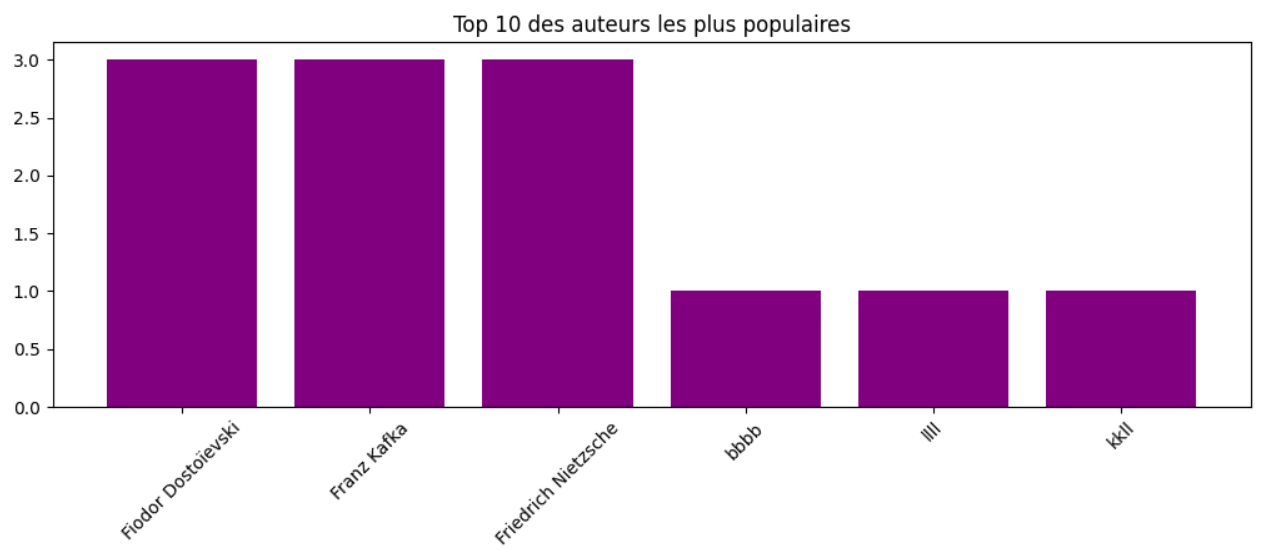
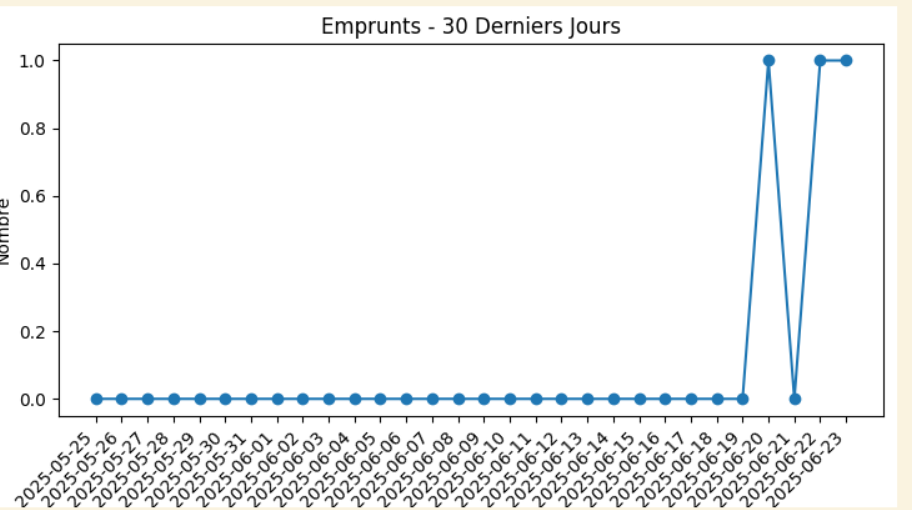
livres = [Livre(\*ligne.strip().split(";")) for ligne in lignes]

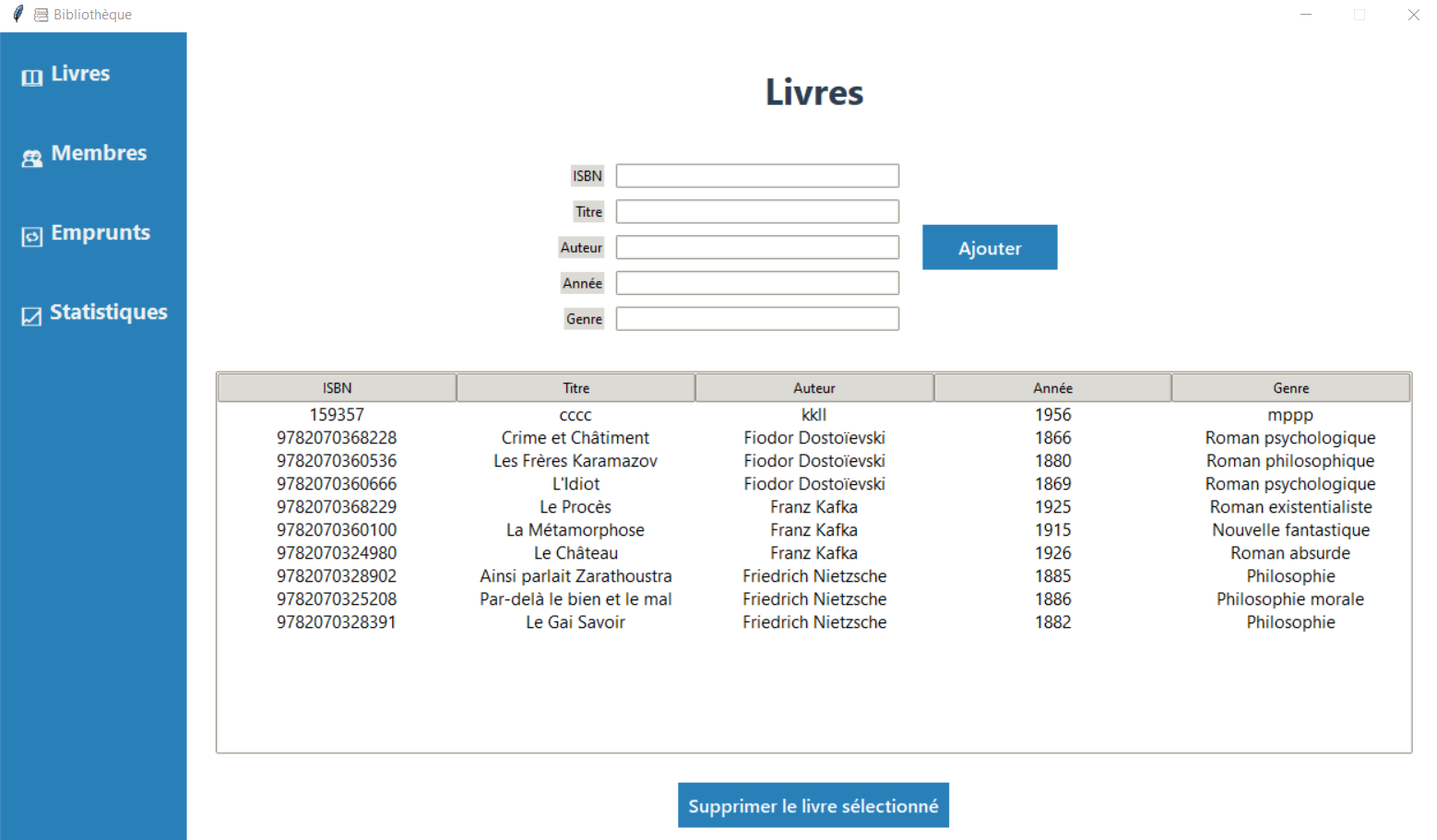
Lors de la sauvegarde, on convertit chaque objet en ligne texte :

for livre in livres:

f.write(f"{livre.ISBN};{livre.titre};{livre.auteur};...\n")

# Captures d’ecran des visualisations :

1. **Diagramme circulaire:% Des livres par genre**
2. **Histogramme:Top 10 des auters les plus populaires:**
3. **Courbe temporelle:Activité des emprunts (30 derniers jours)**
4. **Interface de Bibliotheque**



# Problemes et les solutions :

**• Problème : Majuscules / Espaces dans le debut les titres**

Par exemple : "Le Petit Prince" ≠ " le petit prince "

**Solution** : utilisation de .**lower().strip()** partout où un titre est comparé ou recherché.

**• Problème : Graphes qui s’affichent dans une autre fenêtre**

Initialement, plt.show() ouvrait une fenêtre séparée.

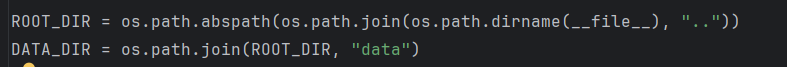
**Solution** : remplacement par **FigureCanvasTkAgg(fig, master=...)** pour afficher les graphes dans l'onglet **Statistiques**.

**Problème :Accès aux fichiers dans différents dossiers**

**Problème rencontré :**  
Lorsqu’on lançait l’application depuis un autre répertoire (ex. : src/interface\_main.py), les chemins relatifs vers les fichiers comme data/livres.txt ne fonctionnaient pas. Le programme ne trouvait pas les fichiers car le chemin relatif changeait selon l’endroit d’où le script était exécuté.

**Cause :**  
Les chemins relatifs comme "../data/livres.txt" ne sont pas fiables si l’utilisateur exécute le script depuis un endroit différent (ex. : clic sur le fichier dans un explorateur ou exécution depuis un IDE).

**Solution apportée :**  
Pour rendre le projet plus **portable et robuste**, on a défini le chemin racine (ROOT\_DIR) dynamiquement en utilisant \_\_file\_\_ :



**• Problème : Interface graphique peu attractive**

Design initial très basique, couleurs neutres, alignements approximatifs.

**Solution** :

* Ajout de ttk.Style() pour boutons, onglets, labels.
* Couleurs : #007acc, #005f99, #f0f2f5
* Onglets bien structurés avec pack(fill='both', expand=True)