

*UNIVERSITE MOHAMMED PREMIER*

*ECOLE NATIONALE des SCIENCES APPLIQUEES D’OUJDA*



**FILIERE : Génie Informatique**

**MODULE : Python**

**Conception et Développement d’une Application de gestion des bibliothèque**

Projet réalisé par :

MOUSSA Khaoula

Encadré par :

Mr. HAJA ZAKARIA

**Année universitaire : 2025/2026**

[Introduction 4](#_Toc202046982)

[Diagramme de classes UML 5](#_Toc202046983)

[Explication des algorithmes clés 5](#_Toc202046984)

[Captures d'écran des visualisations 8](#_Toc202046985)

[Difficultés Rencontrées et Solutions dans le Système de Gestion de Bibliothèque : 9](#_Toc202046986)

Table de figure

[Figure 1Diagramme de classe 5](#_Toc202046959)

[Figure 2Interface gestion des Livres 8](#_Toc202046960)

[Figure 3 Interface gestion des Membres 8](#_Toc202046961)

[Figure 4 Interface gestion des Emprunts 9](#_Toc202046962)

[Figure 5 Interface des statistiques 9](#_Toc202046963)

# Introduction

**Introduction**

Ce projet, baptisé **"Système de Gestion de Bibliothèque"**, ambition de développer une application complète en Python, destinée à autоmаtisеr et à améliorer la gestion des tâches habituelles d'un bibliothèque. Plus en détail, le logiciel devra offrir les fоnctiоnnаlités suivantes :

**La gestion des membres** : Inscription, suivi des emprunts en cours, et historique des activités.

* **La gestion des livres** : Ajout, suppression, et suivi des statuts (disponible/emprunté).
* **La gestion des transactions** : Emprunts, retours, avec validation des contraintes métier (ex : quotas d’emprunt).

Dans le cadre de ce projet, on va appliquer les principes de la programmation orientée objet (POO) afin de représenter les principales entités, tels que Livre, Membre et Bibliothèque. Nous développerons également des eхcеptiоns sur mesure pour traiter les erreurs liées au domaine de manière efficace. Les informations seront conservés dans des fichiers du format ou CSV, garantissant ainsi la durabilité des données entre les différentes sessions. Pur conclure, des tаblеаuх de bord statistiques, élaborés grâce à Matplоtlib, fourniront un représentation visuelle claire de l'activité de la bibliothèque, incluant des éléments comme la répartition des genres littéraires et les auteurs les plus on vоgue.Ce rapport détaillera :

1. **L’architecture logicielle** via un diagramme de classes UML annoté.
2. **Les algorithmes clés** (gestion des emprunts, sauvegarde des données).
3. **Les défis techniques** rencontrés et les solutions apportées.
4. **Les résultats** sous forme de captures d’écran et de visualisations.

L’objectif final est de fournir un outil fonctionnel, modulaire, et facilement extensible pour répondre aux besoins des bibliothèques modernes.

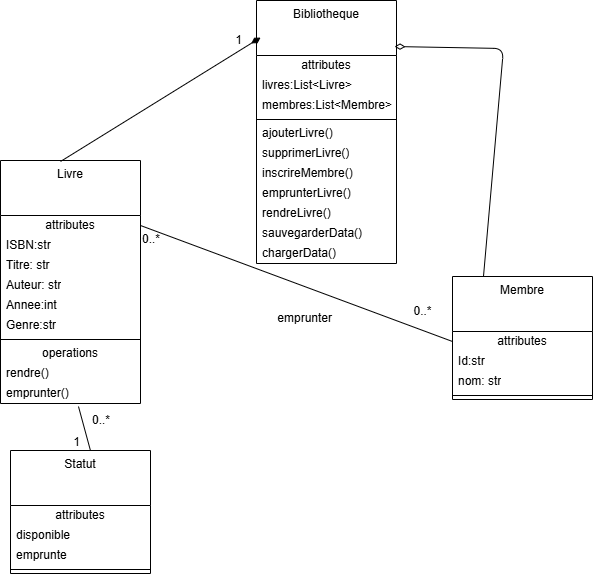
Diagramme de classes UML

Figure 1Diagramme de classe

* Livre : représente un ouvrage, avec ses attributs (ISBN, titre, auteur, etc.) et ses opérations (emprunter, rendre).
* Membre : représente un adhérent, avec un identifiant et un nom.
* Statut : énumération définissant si un livre est disponible ou emprunté.
* Bibliothèque : classe principale contenant les listes de livres et de membres, ainsi que toutes les méthodes de gestion (ajout, suppression, emprunt, retour, sauvegarde).

# Explication des algorithmes clés

**1. Structure de Base**

**Classe BibliothequeApp**

* Fonction : Classe principale de l'interface graphique
* Algorithmes clés :
  + \_\_init\_\_ : Initialise l'application, charge les données et crée les widgets
  + CreateWidgets : Construit l'interface avec les onglets principaux
  + SetupStyle : Configure l'apparence des widgets

**2. Gestion des Livres**

**CreateLivresTab()**

* Fonction : Crée l'onglet de gestion des livres
* Algorithmes :
  + Construction d'un formulaire pour ajouter des livres
  + Création d'un Treeview pour afficher la liste des livres
  + Boutons pour ajouter/supprimer des livres

**AjouterLivre()**

* Récupère les données des champs de saisie
* Valide que tous les champs sont remplis
* Vérifie que l'année est numérique
* Crée un nouvel objet Livre
* Ajoute le livre à la bibliothèque
* Actualise l'affichage et sauvegarde

**SupprimerLivre()**

* Vérifie qu'un livre est sélectionné
* Demande confirmation
* Vérifie que le livre n'est pas emprunté
* Supprime le livre de la bibliothèque
* Actualise l'affichage et sauvegarde

**3. Gestion des Membres**

**CreateMembresTab()**

* Fonction : Crée l'onglet de gestion des membres
* Algorithmes :
  + Formulaire d'inscription des membres
  + Liste des membres avec leurs emprunts

**InscrireMembre()**

* Récupère l'ID et le nom du membre
* Vérifie que les champs sont remplis
* Vérifie que l'ID n'existe pas déjà
* Ajoute le membre à la bibliothèque
* Actualise l'affichage et sauvegarde

**4. Gestion des Emprunts**

**CreateEmpruntsTab()**

* Fonction : Crée l'onglet de gestion des emprunts
* Algorithmes :
  + Section pour emprunter un livre
  + Section pour rendre un livre
  + Combobox dynamiques pour sélectionner membres/livres

**EmprunterLivre()**

* Récupère le membre et le livre sélectionnés
* Vérifie que le livre est disponible
* Vérifie que le membre n'a pas dépassé son quota (5 livres)
* Met à jour le statut du livre
* Ajoute le livre à la liste des emprunts du membre
* Enregistre dans l'historique
* Actualise les affichages

**RendreLivre()**

* Récupère le membre et le livre sélectionnés
* Vérifie que le livre est bien emprunté par ce membre
* Met à jour le statut du livre
* Retire le livre de la liste des emprunts du membre
* Enregistre dans l'historique
* Actualise les affichages

**5. Statistiques**

**CreateStatsTab()**

* Fonction : Affiche des statistiques visuelles
* Algorithmes :
  + Graphique circulaire des genres de livres
  + Graphique des 10 auteurs les plus populaires
  + Courbe des emprunts sur 30 jours

**GenererGraphiqueGenres()**

* Compte le nombre de livres par genre
* Crée un diagramme circulaire avec matplotlib

**GenererGraphiqueAuteurs()**

* Compte le nombre de livres par auteur
* Trie pour obtenir le top 10
* Crée un diagramme en barres

**GenererCourbeEmprunts()**

* Prépare les dates des 30 derniers jours
* Compte les emprunts par date
* Trace une courbe d'évolution

**6. Persistance des Données**

**ChargerData()**

* Charge les livres depuis 'data/livres.txt'
* Charge les membres depuis 'data/membres.txt'
* Charge l'historique depuis 'data/historique.csv'
* Gère les relations entre membres et livres empruntés

**sauvegarderData()**

* Sauvegarde les livres dans 'data/livres.txt'
* Sauvegarde les membres dans 'data/membres.txt'
* Sauvegarde l'historique dans 'data/historique.csv'

**7. Algorithmes Utilitaires**

**ActualiserLivres()**

* Met à jour l'affichage de la liste des livres
* Trie les livres par titre

**ActualiserMembres()**

* Met à jour l'affichage de la liste des membres
* Trie les membres par nom
* Affiche les livres empruntés

**remplir\_comboboxes()**

* Met à jour les listes déroulantes pour les emprunts/retours
* Filtre les livres disponibles pour l'emprunt
* Filtre les membres ayant des emprunts pour les retours

**Gestion des Erreurs**

Le code utilise des exceptions personnalisées pour gérer les cas d'erreur :

* LivreInexistantError : Livre non trouvé
* LivreIndisponible : Livre déjà emprunté
* MembreInexistantError : Membre non trouvé
* QuotaEmpruntDepasseError : Membre a déjà 5 livres

Ce système permet une gestion complète d'une bibliothèque avec interface graphique, persistance des données et visualisation statistique.

Captures d'écran des visualisations**:**

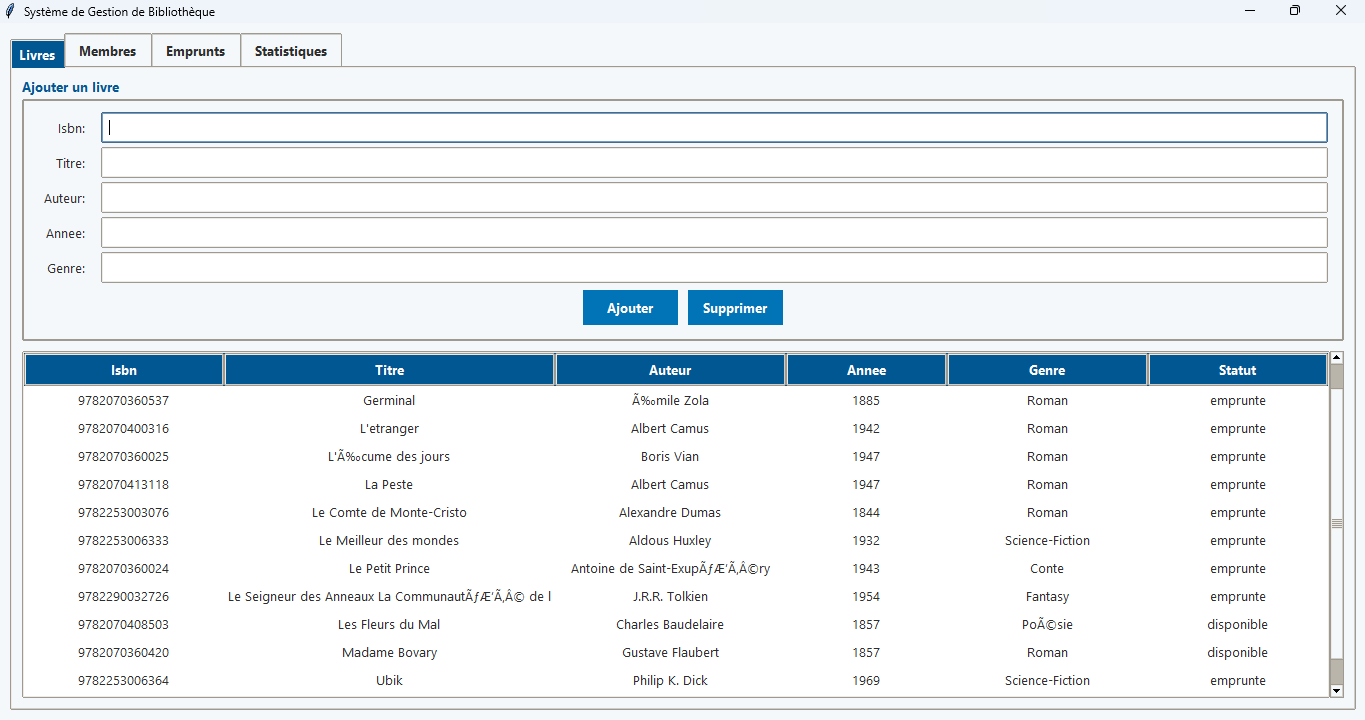


Figure 2Interface gestion des Livres

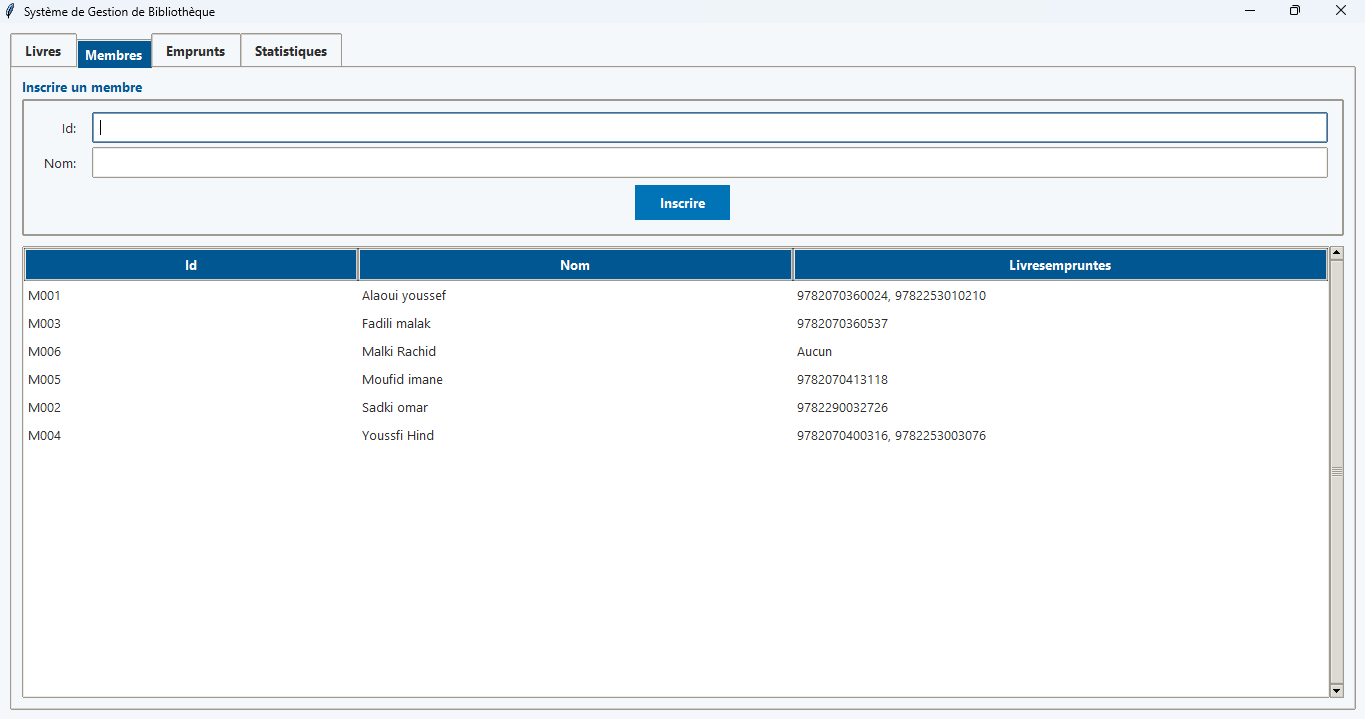


Figure 3 Interface gestion des Membres

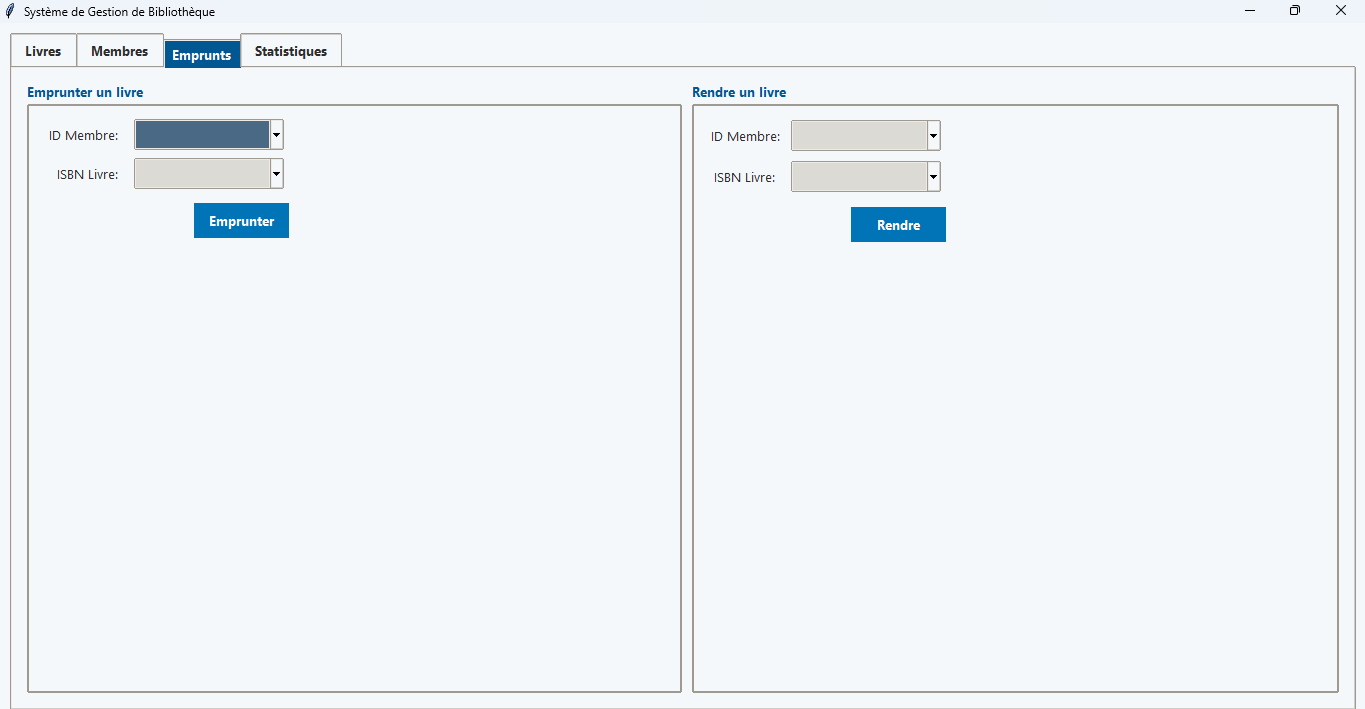


Figure 4 Interface gestion des Emprunts

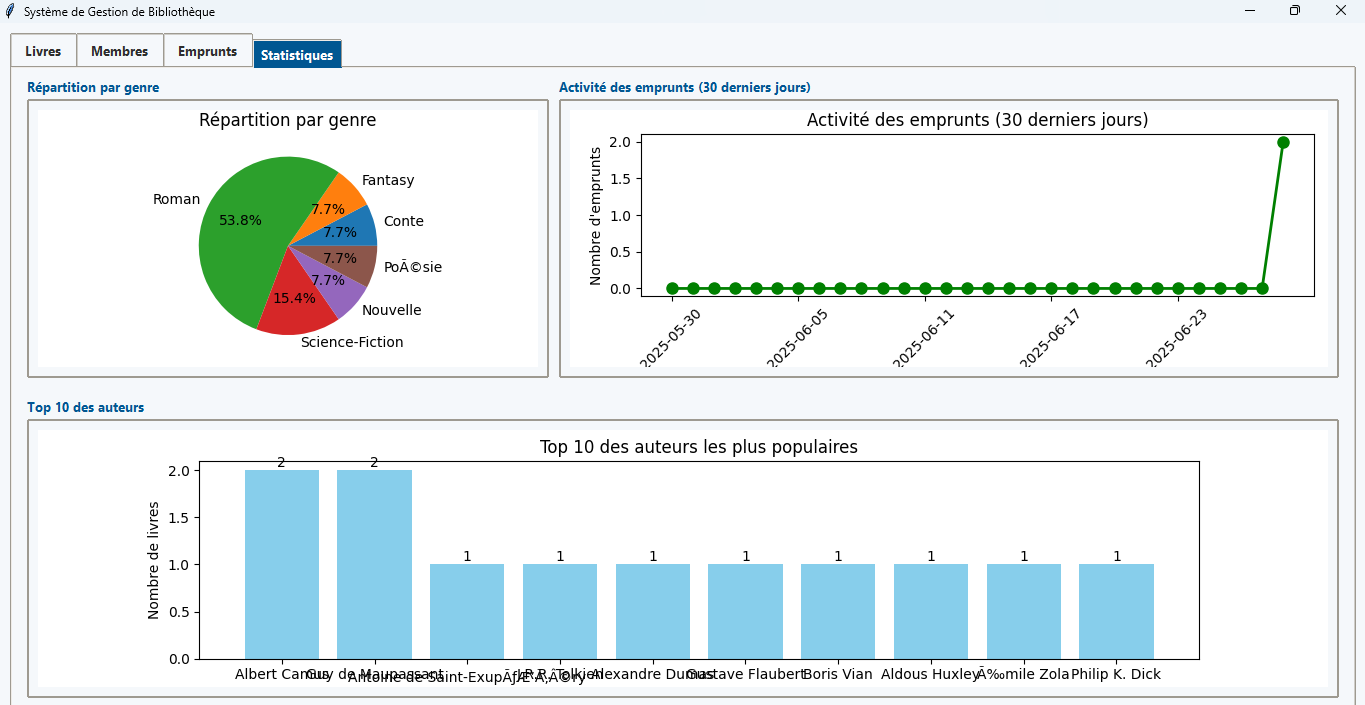


Figure 5 Interface des statistiques

# Difficultés Rencontrées et Solutions dans le Système de Gestion de Bibliothèque :

Voici les principales difficultés que j'ai rencontrées lors du développement de ce système de gestion de bibliothèque, ainsi que les solutions que j'ai mises en place pour les résoudre :

**Défi Initial : Conception d'Interface Complexe**

**Problème** :

* J'ai commencé par utiliser **Figma** pour concevoir des maquettes professionnelles
* Puis j'ai essayé **Tkinter Designer** pour convertir ces maquettes en code Tkinter

**🡺**Le délai d'une semaine s'est avéré insuffisant pour maîtriser ces outils et implémenter complètement l'interface

**1. Gestion des Données et Persistance**

Difficulté :

* Synchronisation entre les objets en mémoire et les fichiers de sauvegarde
* Formatage correct des données lors de la sérialisation/désérialisation
* Gestion des relations entre livres et membres (emprunts)

Solutions :

**Format standardisé** : J'ai utilisé un format CSV avec des séparateurs ";" pour les fichiers texte.

**Chargement relationnel** : Lors du chargement des membres, je reconstruis les relations avec les livres empruntés

**2. Gestion des Emprunts et Retours**

**Difficulté** :

* Vérification des conditions préalables (disponibilité, quota)
* Mise à jour cohérente de tous les états concernés
* Historisation des actions

**Solutions** :

* **Exceptions spécifiques** : Création d'exceptions personnalisé pour chaque cas d'erreur

**3**. **Gestion des Dates et Historique**

**Difficulté** :

* Formatage des dates pour l'affichage et le stockage
* Calcul des périodes pour les statistiques temporelles
* Tri chronologique des données

**Solutions** :

* **Format standard** : Utilisation du format ISO pour le stockage