

## Introduction:

Ce mini-projet a pour objectif de développer une application complète de gestion de bibliothèque en Python, en appliquant des concepts avancés de la programmation orientée objet, la gestion des exceptions, la visualisation de données et la persistance des données.

## 1 ° ) l' architecture du systeme

L'architecture technique de ce projet de gestion de bibliothèque est structurée en une logique métier séparée de l'interface utilisateur :

**1.Modèle (bibliotheque.py):** contient les classes métier à savoir Livre, Membre, et Bibliotheque.

**Livre:** information sur le livre plus la gestion de l'état d'un livre (disponible, emprunté).

**Membre:** Représente un membre (ID, nom) plus la liste de ses livres empruntés la gestion des emprunts au maximum 3 livres.

**Bibliotheque:** le stockage de la liste de livres et de membres, la recherche (trouver\_livre, trouver\_membre), la gestion des Emprunts / retours avec historique (historique.csv) enfin le chargement et sauvegarde des données (livres.json, membres.json).

**2.Contrôleur(main.py):** sert de contrôleur, avec un menu textuel, pour relier les entrées utilisateur aux actions sur les objets métier.

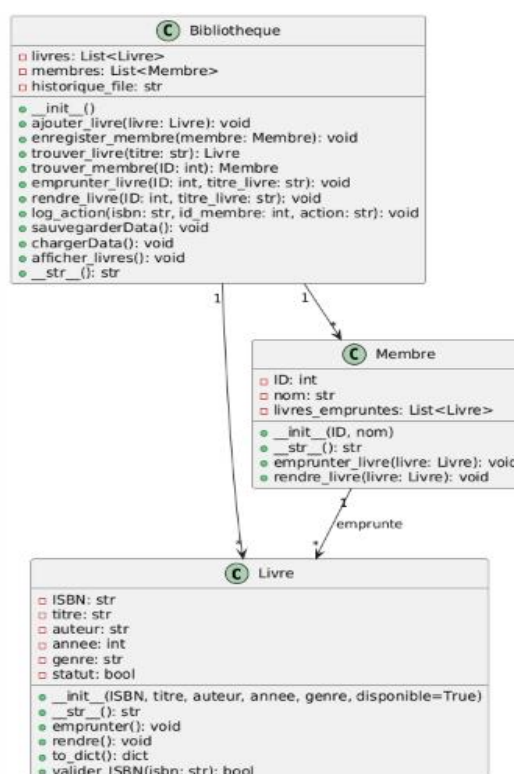
**main.py:** joue le rôle d' interface utilisateur (console) avec un menu texte interactif, permet la validation des entrées (ISBN, ID membre, etc.) et Appels aux méthodes de Bibliotheque.

**exceptions.py:** Définit des exceptions personnalisées utilisées pour gérer les cas d' erreurs métier (telles que LivreIndisponibleError, QuotaEmpruntDepasseError, MembreInexistantError, LivreInexistantError).

**3.Visualisation (visualisation.py):** pour afficher les statistiques graphiques comme le pourcentage des livres par genre, les 10 premiers auteurs les plus populaires ainsi que les activites des emprunts (pendant 30 derniers jours).

## Conception du système

Ce diagramme UML montre les classes Livre, Membre, Bibliothèque et leurs relations.



## 2. Fonctionnement du systeme

Les fonctionnalités principales sont celle d'emprunt qui permet à un membre, identifié par son ID, d'emprunter un livre disponible en vérifiant son quota d'emprunts, en changeant l'état du livre et en enregistrant l'opération dans un fichier historique, et celle de retour permet à un membre de restituer un livre précédemment emprunté, en vérifiant qu'il en est bien le détenteur, en rétablissant l'état du livre à disponible et en enregistrant l'action dans l'historique.

Le menu offre une interface utilisateur en ligne de commande permettant d'accéder aux principales fonctionnalités de gestion de la bibliothèque, comme l'ajout de livres, l'inscription de membres, l'emprunt, le retour, l'affichage et la sauvegarde des données.

```
=== GESTION BIBLIOTHÈQUE ===
1. Ajouter un livre
2. Inscrire un membre
3. Emprunter un livre
4. Rendre un livre
5. Lister tous les livres
6. Afficher les statistiques
7. Sauvegarder et quitter
Votre choix : 
```

### L'algorithme d'emprunt :

L'utilisateur choisit « Emprunter un livre ». `main.py` demande l'ID membre et le titre par l'appel de `biblio.trouver_membre()` et `biblio.trouver_livre()`. Puis `Bibliotheque.emprunter_livre()` appelle `Membre.emprunter_livre()` qui a son tour appelle `Livre.emprunter()`. Un enregistrement de cet emprunt est mémorisé dans `historique.csv`. Enfin, Le tout peut être sauvegardé dans les fichiers `livres.json` pour les données des livres et `membres.json` pour les données des membres

<pre>=== GESTION BIBLIOTHÈQUE === 1. Ajouter un livre 2. Inscrire un membre 3. Emprunter un livre 4. Rendre un livre 5. Lister tous les livres 6. Afficher les statistiques 7. Sauvegarder et quitter Votre choix : 5 Liste des livres dans la bibliothèque : Livre(ISBN=978-2-07-036822-8, titre=Lady Lazarus, auteur=Sylvia Plath, annee=1965, genre=Roman, état=emprunté)</pre>	<pre>=== GESTION BIBLIOTHÈQUE === 1. Ajouter un livre 2. Inscrire un membre 3. Emprunter un livre 4. Rendre un livre 5. Lister tous les livres 6. Afficher les statistiques 7. Sauvegarder et quitter Votre choix : 3 ID du membre : 1 Titre du livre à emprunter : Lady Lazarus</pre>
<pre>=== GESTION BIBLIOTHÈQUE === 1. Ajouter un livre 2. Inscrire un membre 3. Emprunter un livre 4. Rendre un livre 5. Lister tous les livres 6. Afficher les statistiques 7. Sauvegarder et quitter Votre choix : 5 Liste des livres dans la bibliothèque : Livre(ISBN=978-2-07-036822-8, titre=Lady Lazarus, auteur=Sylvia Plath, annee=1965, genre=Roman, état=disponible) Livre(ISBN=978-0-14-044926-6, titre=L'Étranger, auteur=Albert Camus, annee=1942, genre=Roman, état=disponible)</pre>	<pre>=== GESTION BIBLIOTHÈQUE === 1. Ajouter un livre 2. Inscrire un membre 3. Emprunter un livre 4. Rendre un livre 5. Lister tous les livres 6. Afficher les statistiques 7. Sauvegarder et quitter Votre choix : 7 Données sauvegardées. Au revoir !</pre>

## 4°) Gestion des erreurs (exceptions)

Pour la gestion d'erreurs, on a distingué les exceptions métier (liées à la logique de bibliothèque) aux exceptions de saisie (liées aux entrées utilisateur).

Les **exceptions personnalisées** (liées à la logique de bibliothèque) :

LivreIndisponibleError : levée lorsqu'un utilisateur tente d'emprunter un

QuotaEmpruntDepasseError : levée lorsqu'un membre essaie d'emprunter un livre alors qu'il a déjà atteint le nombre maximal d'emprunts autorisés.

MembreInexistantError : levée lorsqu'un membre avec l'ID spécifié est introuvable dans la bibliothèque.

LivreInexistantError : levée lorsqu'un livre avec le titre donné est introuvable dans la bibliothèque.

**Exceptions standards** (contrôle de saisie dans `main.py`) :

## ValueError :

Lorsqu'un champ requis (titre, auteur, année, etc.) est vide ou invalide.

Lors de la conversion d'un identifiant de membre ou d'une année en entier (int).

Utilisée dans la fonction `verifierEntierPositif(i)` pour rejeter les valeurs non numériques ou négatives.

Vérifier que le code ISBN saisi est valide avant d'ajouter un livre à la bibliothèque.

## Exception (générique) :

Utilisée pour capturer toute autre erreur lors de l'ajout d'un livre ou d'un membre, ou dans les appels à des fonctions métier.

Exemple : tentative d'ajout d'un livre avec un ISBN déjà existant ou tentative de rendre un livre non emprunté.

<pre>=== GESTION BIBLIOTHEQUE === 1. Ajouter un livre 2. Inscrire un membre 3. Emprunter un livre 4. Rendre un livre 5. Lister tous les livres 6. Afficher les statistiques 7. Sauvegarder et quitter Votre choix : 1 ISBN : 0 Erreur : ISBN invalide.</pre>	<pre>=== GESTION BIBLIOTHEQUE === 1. Ajouter un livre 2. Inscrire un membre 3. Emprunter un livre 4. Rendre un livre 5. Lister tous les livres 6. Afficher les statistiques 7. Sauvegarder et quitter Votre choix : 2 ID du membre : 1 Nom du membre : m Erreur : Un membre avec l'ID 1 existe déjà.</pre>	<pre>=== GESTION BIBLIOTHEQUE === 1. Ajouter un livre 2. Inscrire un membre 3. Emprunter un livre 4. Rendre un livre 5. Lister tous les livres 6. Afficher les statistiques 7. Sauvegarder et quitter Votre choix : 3 ID du membre : 1 Titre du livre à emprunter : Lady Lazarus Traceback (most recent call last):   File "c:\Users\fujitsu\Desktop\pojet.f\Work26Juin\BiblioMenu\main.py", line 141, in &lt;module&gt;     3, in emprunter       raise LivreIndisponibleError("Le livre est déjà emprunté.") NameError: name 'LivreIndisponibleError' is not defined C:\Users\fujitsu\Desktop\pojet.f\Work26Juin\BiblioMenu&gt;</pre>
<pre>=== GESTION BIBLIOTHEQUE === 1. Ajouter un livre 2. Inscrire un membre 3. Emprunter un livre 4. Rendre un livre 5. Lister tous les livres 6. Afficher les statistiques 7. Sauvegarder et quitter Votre choix : 1 ISBN : 978-2-07-036822-8 Titre : Erreur lors de l'ajout du livre : le titre ne peut pas être vide.</pre>	<pre>=== GESTION BIBLIOTHEQUE === 1. Ajouter un livre 2. Inscrire un membre 3. Emprunter un livre 4. Rendre un livre 5. Lister tous les livres 6. Afficher les statistiques 7. Sauvegarder et quitter Votre choix : 2 ID du membre : -5 Erreur lors de l'ajout d'un membre : doit être un nombre positif</pre>	
<pre>=== GESTION BIBLIOTHEQUE === 1. Ajouter un livre 2. Inscrire un membre 3. Emprunter un livre 4. Rendre un livre 5. Lister tous les livres 6. Afficher les statistiques 7. Sauvegarder et quitter Votre choix : 4 ID du membre : 1 Titre du livre à emprunter : En attendant Godot Traceback (most recent call last):   File "c:\Users\fujitsu\Desktop\pojet.f\Work26Juin\BiblioMenu\bibliotheque.py", line 141, in rendre_livre     raise Exception("Ce livre n'a pas été emprunté par ce membre.") Exception: Ce livre n'a pas été emprunté par ce membre. C:\Users\fujitsu\Desktop\pojet.f\Work26Juin\BiblioMenu&gt;</pre>	<pre>=== GESTION BIBLIOTHEQUE === 1. Ajouter un livre 2. Inscrire un membre 3. Emprunter un livre 4. Rendre un livre 5. Lister tous les livres 6. Afficher les statistiques 7. Sauvegarder et quitter Votre choix : 3 ID du membre : 1 Titre du livre à emprunter : ppp Traceback (most recent call last):   File "c:\Users\fujitsu\Desktop\pojet.f\Work26Juin\BiblioMenu\main.py", line 141, in &lt;module&gt;     3, in trouver_livre       raise LivreInexistantError(f"Le livre '{titre}' introuvable.") NameError: name 'LivreInexistantError' is not defined C:\Users\fujitsu\Desktop\pojet.f\Work26Juin\BiblioMenu&gt;</pre>	

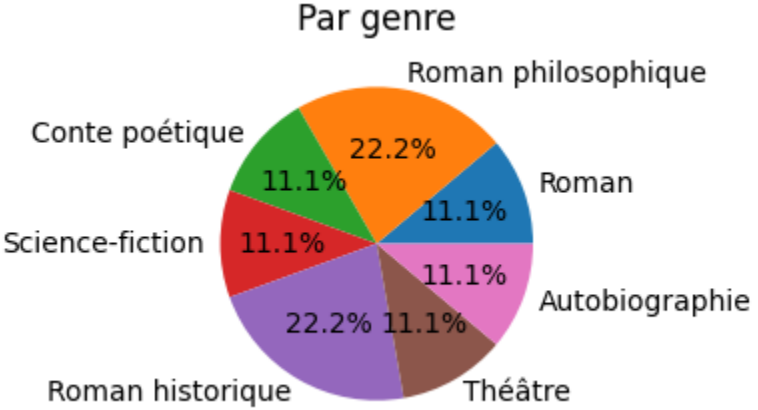
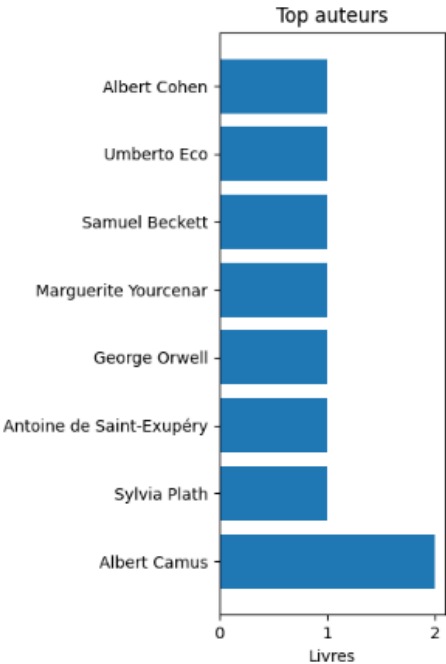
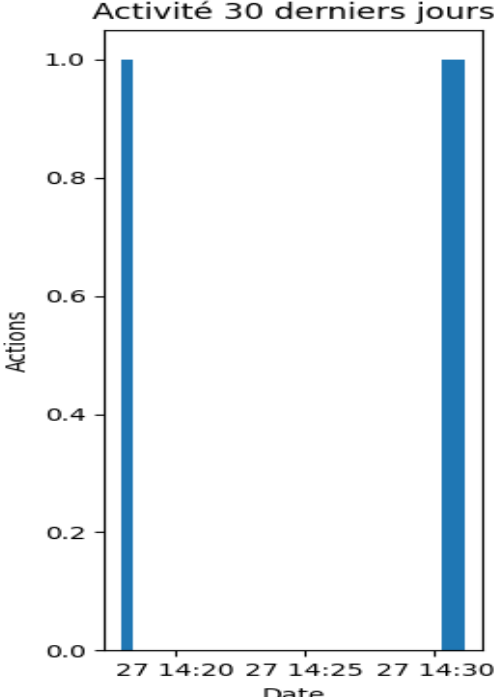
## 5 ° ) Gestion des données:

<pre>{ } livres.json X C: &gt; Users &gt; fujitsu &gt; Desktop &gt; pojet.f &gt; Work2 1 [ 2 { 3   "ISBN": "978-2-07-036822-8", 4   "titre": "Lady Lazarus", 5   "auteur": "Sylvia Plath", 6   "annee": 1965, 7   "genre": "Roman", 8   "disponible": true 9 }, 10 ]</pre> <p>livres.json : stocke les livres au format JSON.</p>	<pre>{ } livres.json X C: &gt; Users &gt; fujitsu &gt; Desktop &gt; pojet.f &gt; Work2 1 [ 2 { 3   "ISBN": "978-2-07-036822-8", 4   "titre": "Lady Lazarus", 5   "auteur": "Sylvia Plath", 6   "annee": 1965, 7   "genre": "Roman", 8   "disponible": true 9 }, 10 { 11   "ISBN": "978-0-14-044926-6", 12   "titre": "L'Étranger", 13   "auteur": "Albert Camus", 14   "annee": 1942, 15   "genre": "Roman philosophique", 16   "disponible": false 17 }</pre> <p>livres.json : stocke les livres au format JSON.</p>	<pre>{ } membres.json X C: &gt; Users &gt; fujitsu &gt; Desktop &gt; pojet.f 1 [ 2 { 3   "ID": 1, 4   "nom": "Mariam", 5   "livres_empruntes": [ 6     "978-0-14-044926-6" 7   ] 8 }, 9 { 10   "ID": 2, 11   "nom": "Manar", 12   "livres_empruntes": [] 13 }</pre> <p>membres.json : stocke les membres + leurs emprunts (par ISBN).</p>
---	---	---

A	B	C	D
6/27/2025 14:17	978-2-07-036822-8		1 emprunt
6/27/2025 14:30	978-2-07-036822-8		1 emprunt
6/27/2025 14:31	978-2-07-036822-8		1 rendu

historique.csv : trace tous les emprunts et rendus (date, ISBN, ID, action).

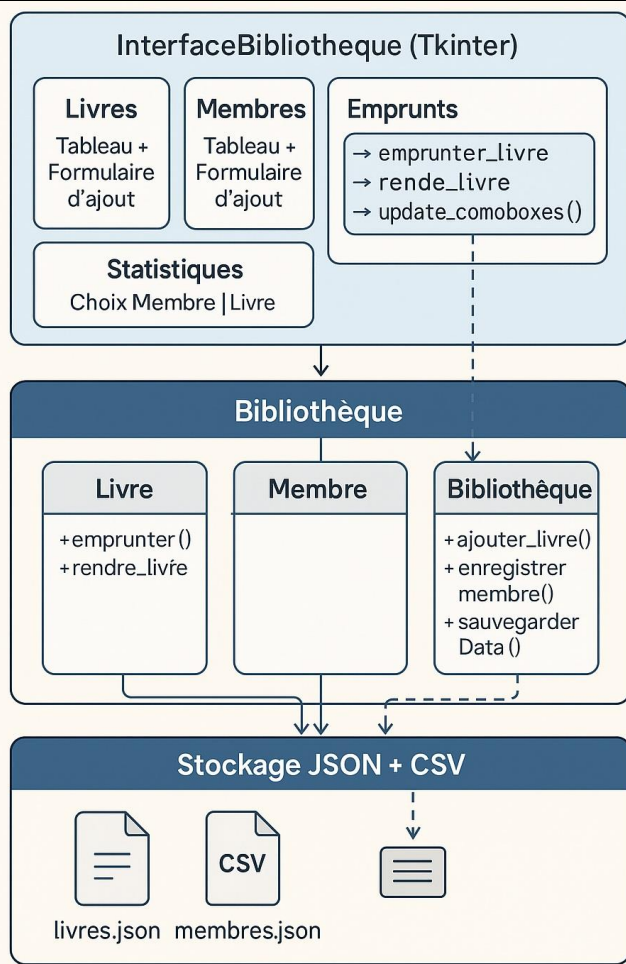
### 6. Visualisation et Statistiques

<p>=== GESTION BIBLIOTHÈQUE ===</p> <ol style="list-style-type: none"> <li>1. Ajouter un livre</li> <li>2. Inscrire un membre</li> <li>3. Emprunter un livre</li> <li>4. Rendre un livre</li> <li>5. Lister tous les livres</li> <li>6. Afficher les statistiques</li> <li>7. Sauvegarder et quitter</li> </ol> <p>Votre choix : 6</p> <p>Suite au choix 6 dans le menu :</p>	<p><b>Diagramme circulaire</b> : Pourcentage des livres par genre</p> <p>Par genre</p>  <table border="1"> <caption>Données du diagramme circulaire</caption> <thead> <tr> <th>Genre</th> <th>Pourcentage</th> </tr> </thead> <tbody> <tr> <td>Roman philosophique</td> <td>22.2%</td> </tr> <tr> <td>Roman</td> <td>11.1%</td> </tr> <tr> <td>Autobiographie</td> <td>11.1%</td> </tr> <tr> <td>Théâtre</td> <td>11.1%</td> </tr> <tr> <td>Roman historique</td> <td>22.2%</td> </tr> <tr> <td>Science-fiction</td> <td>11.1%</td> </tr> <tr> <td>Conte poétique</td> <td>11.1%</td> </tr> </tbody> </table>	Genre	Pourcentage	Roman philosophique	22.2%	Roman	11.1%	Autobiographie	11.1%	Théâtre	11.1%	Roman historique	22.2%	Science-fiction	11.1%	Conte poétique	11.1%										
Genre	Pourcentage																										
Roman philosophique	22.2%																										
Roman	11.1%																										
Autobiographie	11.1%																										
Théâtre	11.1%																										
Roman historique	22.2%																										
Science-fiction	11.1%																										
Conte poétique	11.1%																										
<p><b>Histogramme</b> : Top 10 auteurs les plus empruntés</p> <p>Top auteurs</p>  <table border="1"> <caption>Données de l'histogramme</caption> <thead> <tr> <th>Auteur</th> <th>Nombre de livres</th> </tr> </thead> <tbody> <tr> <td>Albert Cohen</td> <td>1</td> </tr> <tr> <td>Umberto Eco</td> <td>1</td> </tr> <tr> <td>Samuel Beckett</td> <td>1</td> </tr> <tr> <td>Marguerite Yourcenar</td> <td>1</td> </tr> <tr> <td>George Orwell</td> <td>1</td> </tr> <tr> <td>Antoine de Saint-Exupéry</td> <td>1</td> </tr> <tr> <td>Sylvia Plath</td> <td>1</td> </tr> <tr> <td>Albert Camus</td> <td>2</td> </tr> </tbody> </table>	Auteur	Nombre de livres	Albert Cohen	1	Umberto Eco	1	Samuel Beckett	1	Marguerite Yourcenar	1	George Orwell	1	Antoine de Saint-Exupéry	1	Sylvia Plath	1	Albert Camus	2	<p><b>Courbe temporelle</b> : Historique des emprunts sur 30 jours</p> <p>Activité 30 derniers jours</p>  <table border="1"> <caption>Données de la courbe temporelle</caption> <thead> <tr> <th>Date</th> <th>Actions</th> </tr> </thead> <tbody> <tr> <td>27 14:20</td> <td>1.0</td> </tr> <tr> <td>27 14:25</td> <td>0.0</td> </tr> <tr> <td>27 14:30</td> <td>1.0</td> </tr> </tbody> </table>	Date	Actions	27 14:20	1.0	27 14:25	0.0	27 14:30	1.0
Auteur	Nombre de livres																										
Albert Cohen	1																										
Umberto Eco	1																										
Samuel Beckett	1																										
Marguerite Yourcenar	1																										
George Orwell	1																										
Antoine de Saint-Exupéry	1																										
Sylvia Plath	1																										
Albert Camus	2																										
Date	Actions																										
27 14:20	1.0																										
27 14:25	0.0																										
27 14:30	1.0																										

### 3. Interface Utilisateur

Pour encapsuler les règles métier et les données indépendamment de l'interface, on a utilisé les classes Livre, Membre et Bibliothèque définies précédemment dans les fichiers livres.py, membres.py et bibliotheque.py.

Le diagramme ci-contre décrit l'architecture fonctionnelle de notre interface développée grâce à Tkinter.



Contient une classe InterfaceBibliotheque(tk.Tk) :

- Crée une fenêtre principale avec **4 onglets (Notebook)** :
  - **Livres** : tableau + formulaire d'ajout
  - **Membres** : tableau + formulaire d'ajout
  - **Emprunts** : comboBox pour choisir un membre et un livre
  - **Statistiques** : (facultatif pour les futurs ajouts)

**Connexion à la logique métier :**

- Utilise une instance `self.biblio = Bibliotheque()` pour accéder aux livres et membres
- Les **ajouts/emprunts/rendus modifient directement** l'instance biblio
- Les TreeView sont **synchronisés dynamiquement** avec les données métier

#### Synchronisation Interface ↔ Métier ↔ JSON

Action dans l'interface	Impact sur la classe Bibliotheque	Sauvegarde
Ajouter un livre	<code>Bibliotheque.ajouter_livre()</code>	livres.json
Ajouter un membre	<code>Bibliotheque.enregistrer_membre()</code>	membres.json
Emprunter un livre	<code>biblio.emprunter_livre() → Livre.emprunter()</code>	+ historique.csv
Rendre un livre	<code>biblio.rendre_livre() → Livre.rendre()</code>	+ historique.csv
Quitter / bouton save	<code>biblio.sauvegarderData()</code>	écrire dans les fichiers JSON

Les fichiers JSON servent de **stockage persistant**, et sont **chargés au démarrage** via `chargerData()`.

Les figures ci-dessous montrent l'exécution de quelques fonctionnalités de l'interface graphique :

Gestion de Bibliothèque

Livres Membres Emprunts Statistiques

Gestion des Livres

ISBN	Titre	Auteur	Année	Genre	État
978-2-07-036022-8	Lady Lazarus	Sylvia Plath	1965	Roman	Disponible
978-0-14-044026-6	L'Étranger	Albert Camus	1942	Roman philosophique	Emprunté
978-2-253-00514-4	Le Petit Prince	Antoine de Saint-Exupéry	1943	Conte poétique	Disponible
978-2-07-036033-4	La Peste	Albert Camus	1947	Roman philosophique	Emprunté
978-2-356-11156-1	1984	George Orwell	1949	Science-fiction	Emprunté
978-2-07-036002-4	L'Œuvre au noir	Marguerite Yourcenar	1968	Roman historique	Emprunté
978-2-330-07409-7	En attendant Godot	Samuel Beckett	1952	Théâtre	Disponible
978-2-220-11465-2	Le Nom de la rose	Umberto Eco	1980	Roman historique	Emprunté
978-2-253-08703-4	Le Livre de ma mère	Albert Cohen	1954	Autobiographie	Disponible

ISBN

Titre

Auteur

Année

Genre

Ajouter Livre

Activer Windows Accédez aux paramètres

Gestion de Bibliothèque

Livres Membres Emprunts Statistiques

Gestion des Membres

ID	Nom	Livres empruntés
1	Mariam	1
2	Marar	0
3	Abdelghani	2
4	Tasnim	0
5	Youssef	0
6	Amine	0
7	Noor	1
8	Fatima	0
9	Tariq	0
10	Amina	0
11	Omar	0
12	Rane	0
13	Zakaria	1
14	Lina	0
15	Walid	0
16	Aya	0
17	Rachid	0
18	Selma	0
19	Hicham	0
20	Imane	0

ID

Nom

Ajouter Membre

Activer Windows Accédez aux paramètres

Gestion de Bibliothèque

Livres Membres Emprunts Statistiques

Emprunter/Rendre un livre

ID Membre

Titre Livre

Emprunter

Rendre

Emprunter/Rendre un livre

ID Membre

Titre Livre

Emprunter

Rendre

Emprunter/Rendre un livre

ID Membre

Titre Livre

Emprunter

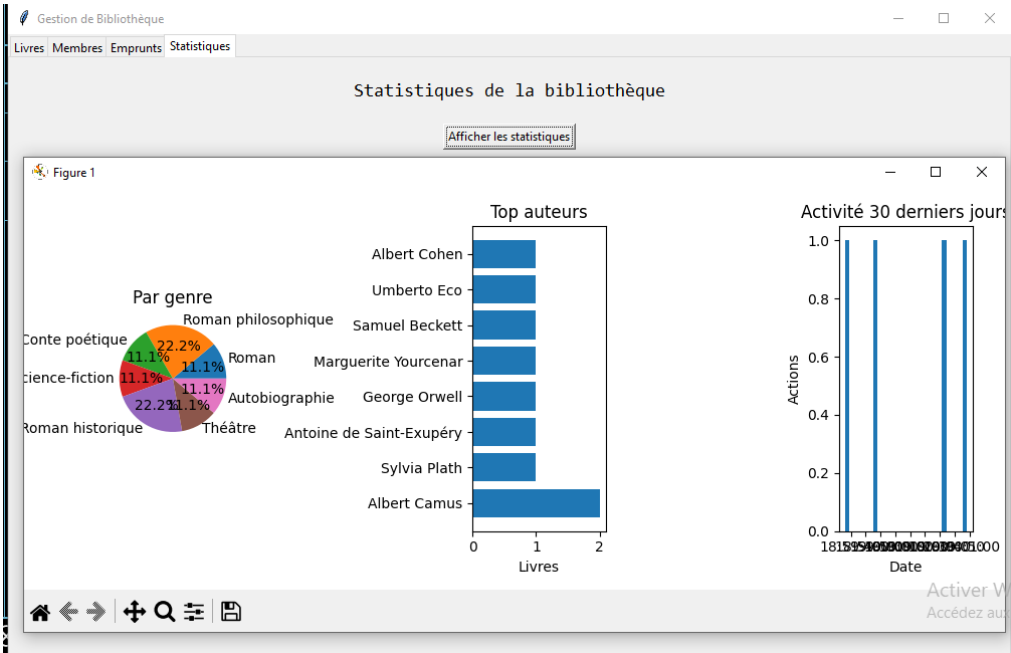
Rendre

Gestion de Bibliothèque

Livres Membres Emprunts Statistiques

Statistiques de la bibliothèque

Afficher les statistiques



## **7. Conclusion**

### **Difficultés rencontrées**

Pendant le développement, plusieurs problèmes ont été rencontrés et résolus :

- La synchronisation entre le “backend” et le “frontend”: l’emploi du modèle MCV a permis une bonne répartition du projet en des fichiers séparés et a garanti le partage des données entre les classes.
- Le contrôle de saisie du code ISBN m’a pris plus de temps causé par l’essai de plusieurs méthodes, heureusement l’emploi de la bibliothèque stdnum était simple et efficace.
- Conflits d’emprunts: Il arrivait que deux utilisateurs essaient d’emprunter le même livre. Pour la solution l’utilisation des exceptions personnalisées était efficace.
- Sauvegarde des listes complexes : Il était difficile d’enregistrer des listes (ex : livres empruntés) dans les fichiers texte. Pour la solution l’utilisation du format JSON est plus adaptée.

Néanmoins, Ce projet m’a permis de renforcer mes compétences en programmation. J’ai amélioré ma maîtrise de la programmation orientée objet, ce qui m’a aidé à mieux structurer le code avec des classes. J’ai aussi appris à gérer les erreurs grâce aux exceptions personnalisées. De plus, j’ai travaillé sur la visualisation des données en créant des graphiques pour analyser la gestion de la bibliothèque. Enfin, ce projet m’a donné une première expérience avec les interfaces graphiques, notamment en utilisant la bibliothèque Tkinter pour créer des fenêtres et des menus interactifs.

### **Annexes**

GitHub : [mariamej7](#)

Vidéo de démonstration :