Compte rendu du projet : Biblio Simple

Application de gestion de bibliothèque avec Tkinter et SQLite

Réalisé par : Yassine Boumezough et Darid Bakr

Le projet Biblio Simple est une application de gestion de bibliothèque réalisée en langage Python, utilisant les bibliothèques Tkinter pour l'interface graphique et SQLite pour la base de données. L'ensemble du code est organisé de manière à séparer clairement la gestion des données, les fonctions métiers et la partie interface utilisateur.

Dès le lancement, l'application crée automatiquement la base de données locale bibliomanager.db si elle n'existe pas encore, puis initialise les tables nécessaires au fonctionnement du programme.

1. Présentation générale et Fonctions principales

Contexte et objectif

Le projet **Biblio Simple** a pour but de développer une application de gestion de bibliothèque simple et pratique. Elle permet d'organiser les livres, les adhérents (membres) et les prêts au sein d'une structure, sans avoir besoin de connaissances avancées en informatique. L'objectif est de fournir une **solution complète mais légère**, qui fonctionne localement sur un ordinateur, sans serveur ni installation complexe. L'utilisateur peut facilement gérer les données via une **interface graphique intuitive**, créée avec **Tkinter**, tout en s'appuyant sur une base de données **SQLite**, intégrée directement dans Python.

Architecture globale

L'application est construite autour de **trois grandes composantes complémentaires** :

1. La base de données (SQLite)

Elle stocke toutes les informations nécessaires au fonctionnement de l'application :

- Les **livres disponibles** : titre, auteur, nombre total d'exemplaires et disponibilité

- Les **membres inscrits** : nom et coordonnées ;

- Les **prêts effectués** : dates d'emprunt et de retour.

Cette base est créée automatiquement lors du **premier lancement** du programme, sous le fichier nommé **bibliomanager.db**, et reste enregistrée localement sur la machine de l'utilisateur.

2. La logique applicative (Python)

Le cœur du projet est développé en langage Python. Il contient un ensemble de fonctions de gestion pour chaque entité du système, Les fonctions sont regroupées en trois catégories principales : la gestion des livres, des membres et des prêts. Chaque fonction interagit directement avec la base de données SQLite et garantit la cohérence des informations.

A. Fonctions liées aux livres

add_book(title, author, total)

```
def add_book(title, author, total):  1 usage
    total = int(total)
    if total < 0:
        raise ValueError("total_copies doit être ≥ 0")
    db.execute( sql: "INSERT INTO Books(title, author, total_copies, available_copies) VALUES(?,?,?,?)",
                parameters: (title.strip(), author.strip(), total, total))
    db.commit()
```

Ajoute un nouveau livre à la base de données., Le nombre d'exemplaires disponibles est initialement égal au nombre total d'exemplaires saisis.

update_book(book_id, title, author, total)

```
def update_book(book_id, title, author, total):  1 usage
    total = int(total)
    cur = db.execute( sql: "SELECT available_copies FROM Books WHERE id=?",  parameters: (book_id,))
    row = cur.fetchone()
    if row is None:
        raise ValueError("Livre introuvable")
    available = row[0]
    if total < available:
        available = total  # garder la cohérence
    db.execute( sql: "UPDATE Books SET title=?, author=?, total_copies=?, available_copies=? WHERE id=?",
                parameters: (title.strip(), author.strip(), total, available, book_id))
    db.commit()
```

Met à jour les informations d'un livre existant. La fonction vérifie que le nombre d'exemplaires disponibles ne dépasse pas le total.

delete_book(book_id)

```
def delete_book(book_id):  1 usage
    cur = db.execute( sql: "SELECT COUNT(*) FROM Loans WHERE book_id=? AND return_date IS NULL",  parameters: (book_id,)
    if cur.fetchone()[0] > 0:
        raise ValueError("Impossible: livre référencé par un prêt en cours.")
    db.execute( sql: "DELETE FROM Books WHERE id=?",  parameters: (book_id,))
    db.commit()
```

Supprime un livre de la base de données uniquement s'il n'est pas référencé par un prêt en cours. Cette vérification empêche la suppression accidentelle de livres encore empruntés.

list_books(search="")

```
def list_books(search=""):  1 usage
    like = f"%{(search or '').strip()}%"
    cur = db.execute(
        sql: "SELECT id, title, author, total_copies, available_copies "
        "FROM Books WHERE title LIKE ? OR author LIKE ? ORDER BY id DESC",
        parameters: (like, like)
    )
    return cur.fetchall()
```

Recherche et affiche la liste des livres correspondant à un mot-clé (titre ou auteur).Si aucun mot-clé n'est fourni, tous les livres sont affichés.

B. Fonctions liées aux membres

add_member(name, phone)

```
def add_member(name, phone):  1 usage
    name = name.strip()
    phone = (phone or "").strip()
    if not name:
        raise ValueError("Nom obligatoire.")
    db.execute( sql: "INSERT INTO Members(name, phone) VALUES(?,?)",  parameters: (name, phone))
    db.commit()
```

Ajoute un nouveau membre à la base de données avec son nom et son numéro de téléphone. Le nom est obligatoire ; le téléphone est facultatif.

update_member(member_id, name, phone)

```
def update_member(member_id, name, phone):  1 usage
    db.execute( sql: "UPDATE Members SET name=?, phone=? WHERE id=?",  parameters: (name.strip(), (phone or "").strip()
    db.commit()
```

Met à jour les informations d'un membre existant (nom et téléphone).

delete_member(member_id)

```
def delete_member(member_id):  1 usage
    cur = db.execute( sql: "SELECT COUNT(*) FROM Loans WHERE member_id=? AND return_date IS NULL",
    if cur.fetchone()[0] > 0:
        raise ValueError("Impossible: membre avec prêt en cours.")
    db.execute( sql: "DELETE FROM Members WHERE id=?",  parameters: (member_id,))
    db.commit()
```

Supprime un membre uniquement s'il n'a pas de prêt en cours. Cette vérification évite de supprimer un adhérent qui possède encore des livres non retournés.

list_members(search="")

```
def list_members(search=""):  1 usage
    like = f"%{(search or '').strip()}%"
    cur = db.execute(
        sql: "SELECT id, name, phone FROM Members "
        "WHERE name LIKE ? OR IFNULL(phone,'') LIKE ? ORDER BY id DESC"
        parameters: (like, like)
    )
    return cur.fetchall()
```

Recherche un membre à partir de son nom ou de son numéro de téléphone. Si le champ de recherche est vide, la liste complète des membres est renvoyée.

C. Fonctions liées aux prêts et interface

Les fonctions liées aux **prêts** assurent toute la gestion des emprunts et des retours de livres au sein de la bibliothèque. Elles permettent d'enregistrer un prêt en vérifiant d'abord la disponibilité du livre, puis en insérant un nouvel enregistrement dans la table correspondante et en diminuant le nombre d'exemplaires disponibles. Lorsqu'un livre est retourné, une fonction met à jour la date de retour et rétablit le stock du livre concerné. D'autres fonctions permettent de consulter la liste complète des prêts en cours, avec les informations sur le membre, le livre, la date d'emprunt et la date d'échéance. Quant aux **fonctions liées à l'interface**, elles gèrent l'affichage et le comportement de la fenêtre graphique créée avec Tkinter. Elles permettent de centrer la fenêtre sur l'écran, de fermer correctement la connexion à la base de données lors de la fermeture du programme, et de lancer l'application en initialisant les trois onglets principaux (Livres, Membres, Prêts). Ces fonctions garantissent une expérience utilisateur fluide et un fonctionnement stable de l'application.

2. Base de données et structure

La base de données utilisée dans **Biblio Simple** repose sur le moteur **SQLite**, intégré directement à Python. Ce choix permet d'avoir un système léger, rapide et autonome, sans besoin d'un serveur externe.

Les données sont enregistrées localement dans un fichier nommé **bibliomanager.db**, créé automatiquement au premier lancement du programme.

L'ensemble de la structure repose sur trois tables principales :

Books, Members et Loans, qui sont reliées entre elles par des clés étrangères afin de garantir la cohérence des informations.

3. Interface graphique

L'interface graphique de **Biblio Simple** a été réalisée avec la bibliothèque **Tkinter**, fournie nativement avec Python. Elle constitue le point d'interaction principal entre l'utilisateur et la base de données, en rendant la gestion des livres, des membres et des prêts **simple, rapide et intuitive**. Cette fenêtre contient un **cahier à trois onglets** (Notebook Tkinter), chacun dédié à une fonctionnalité précise

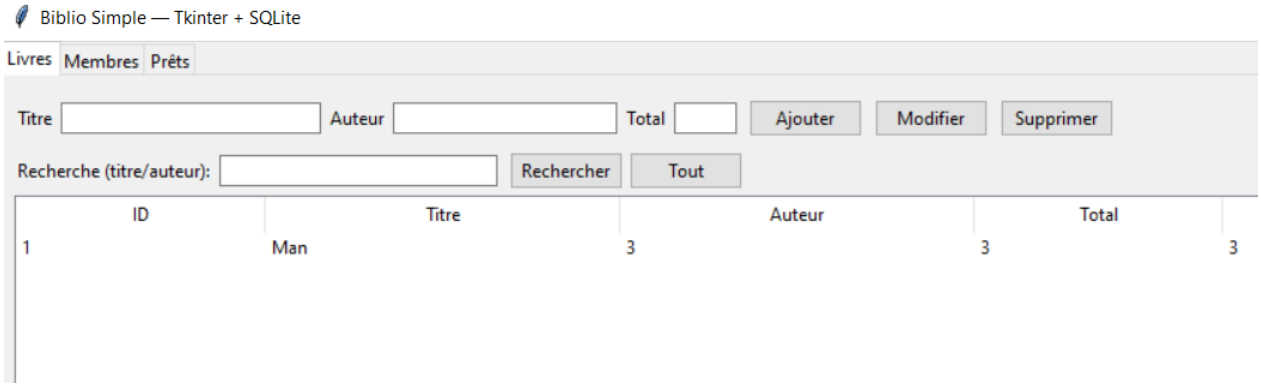
L'onglet **Livres** permet de gérer l'ensemble du catalogue de la bibliothèque. Il comprend des zones de texte pour saisir le **titre**, l'**auteur** et le **nombre d'exemplaires**, ainsi que plusieurs boutons

Ajouter : enregistre un nouveau livre dans la base ;

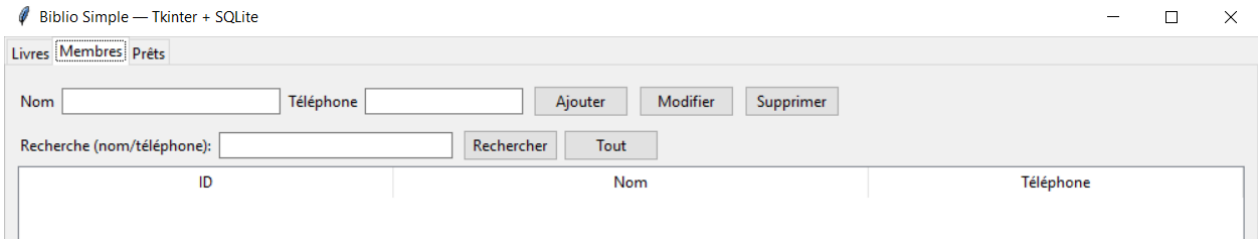
Modifier : met à jour les informations du livre sélectionné ;

Supprimer : retire un livre de la base après confirmation ;

Rechercher / Tout : filtre les résultats selon un mot-clé (titre ou auteur) ou réinitialise la liste.

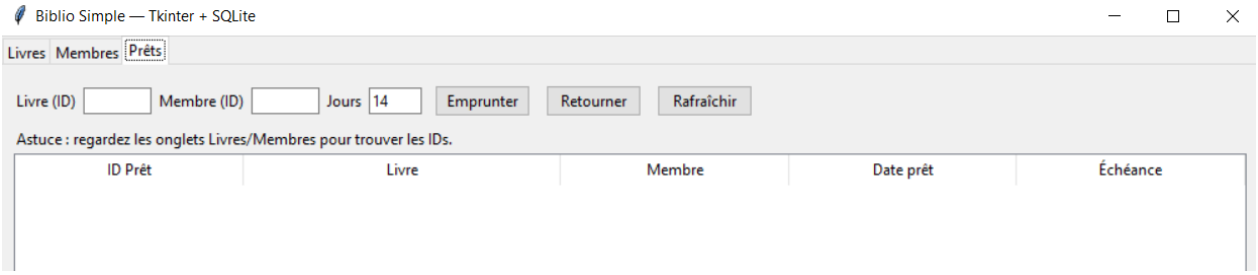


L'onglet **Membres** est consacré à la gestion des adhérents de la bibliothèque. Il propose des champs pour le **nom** et le **numéro de téléphone**, ainsi que des boutons similaires



L'onglet **Prêts** permet de gérer les emprunts et les retours de livres. L'utilisateur saisit l'**ID du livre**, l'**ID du membre** et la **durée du prêt (en jours)**, puis sélectionne l'action souhaitée :

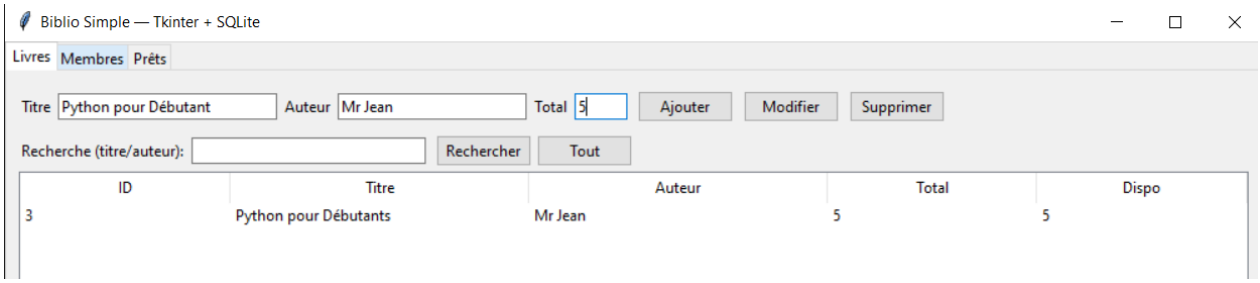
- **Emprunter** : enregistre un nouveau prêt et diminue le nombre d'exemplaires disponibles du livre choisi ;
- **Retourner** : marque un prêt comme retourné et remet l'exemplaire dans le stock ;
- **Rafraîchir** : met à jour la liste des prêts en cours.



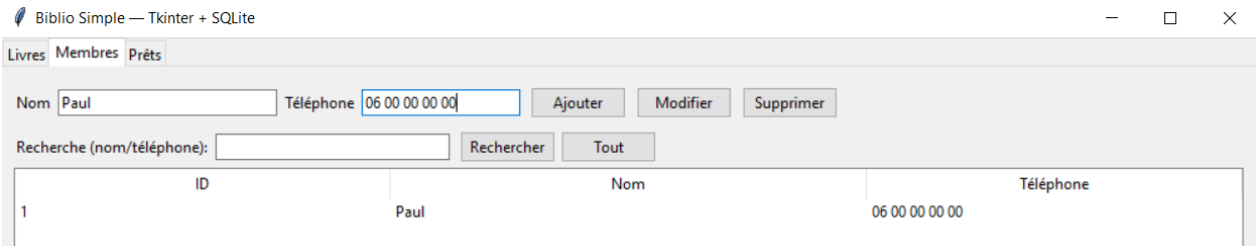
4. Exemple d'utilisation

Lors du lancement, l'application crée automatiquement la base de données `bibliomanager.db` si elle n'existe pas. Les tables sont alors initialisées et prêtes à être utilisées. Exemple de scénario :

1. L'utilisateur ajoute un livre intitulé "Python pour Débutants" de l'auteur Mr Jean(5 exemplaires).

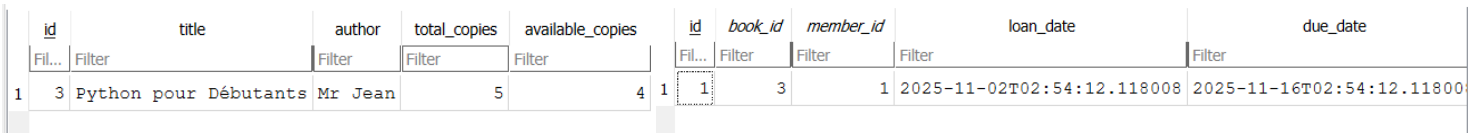


2. Il inscrit un nouveau membre "Paul". Son numéro de tel : 06 00 00 00 00



3. Il enregistre un prêt du livre à Paul pour 14 jours.

4. Lors du retour du livre, il marque le prêt comme "retourné". Toutes ces opérations mettent à jour automatiquement la base de données.



	id	title	author	total_copies	available_copies
	Fil...	Filter	Filter	Filter	Filter
1	3	Python pour Débutants	Mr Jean	5	4

	id	book_id	member_id	loan_date	due_date
	Fil...	Filter	Filter	Filter	Filter
1	1	3	1	2025-11-02T02:54:12.118008	2025-11-16T02:54:12.11800

	id	name	phone
	Fil...	Filter	Filter
1	1	Paul	06 00 00 00 00

5. Conclusion + Code Source

Le projet **Biblio Simple** illustre comment Python peut être utilisé pour créer une application complète de gestion en s'appuyant sur des outils simples qui sont Tkinter et SQLite. Le code source est organisé de manière structurée et chaque section est commentée, ce qui facilite sa compréhension et sa maintenance. Cette organisation permet de suivre facilement la logique de l'application et de comprendre le rôle de chaque fonction. La solution est légère, portable et facilement adaptable, constituant une base idéale pour un projet de bibliothèque ou de gestion de stock à petite échelle. L'ensemble du code est accessible publiquement sur GitHub à l'adresse suivante : https://github.com/YassineBoumezough/biblioth-que_manager.git