RAPPORT DE PROJET OOP:

APPLICATION DE GESTION D'UNE CINEMA

réalisée par:
AYA SMALI
IBTISSAME EL-JEZZAR

Encadre par:
PR.Mustapha HAIN

SOMMAIRE

INTRODUCTION

Dans le cadre du module de Python avancé, il nous a été proposé de réaliser un projet pratique visant à mettre en œuvre nos connaissances acquises en programmation et en conception d'applications.

Le thème retenu pour ce projet est la gestion d'un cinéma, un sujet pertinent qui permet d'explorer de manière concrète la relation entre la gestion de données et l'interaction utilisateur via une interface graphique.
L'objectif principal de ce travail est de concevoir une application capable d'assurer la gestion complète des informations relatives aux films, aux clients, ainsi qu'aux réservations.
Pour cela, nous avons utilisé le langage Python, accompagné du module Tkinter pour la partie interface graphique et de SQLite3 pour la gestion de la base de données.
Ce projet nous a permis de comprendre comment relier efficacement une interface utilisateur à une base de données, tout en garantissant la simplicité, la convivialité et la fiabilité du système.
Ainsi, cette application illustre l'importance de la programmation événementielle et de la structuration des données dans le développement d'outils de gestion modernes.

OBJECTIFS DU PROJET

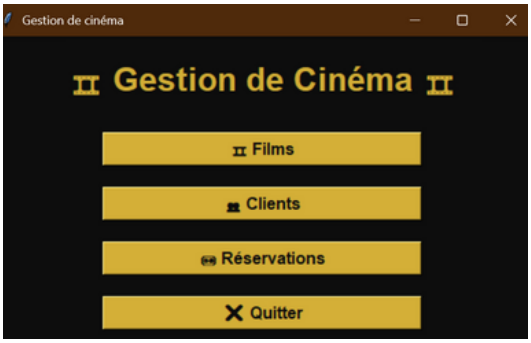
- CRÉER UNE BASE DE DONNÉES POUR STOCKER ET ORGANISER LES INFORMATIONS RELATIVES AUX CLIENTS, AUX FILMS ET AUX RÉSERVATIONS.
- DÉVELOPPER UNE INTERFACE GRAPHIQUE CONVIVIALE AVEC LE MODULE TKINTER, FACILITANT LA MANIPULATION DES DONNÉES SANS PASSER PAR DES COMMANDES SQL.
- PERMETTRE LES OPÉRATIONS DE GESTION COURANTES, TELLES QUE L'AJOUT, LA SUPPRESSION ET L'AFFICHAGE DES INFORMATIONS POUR CHAQUE TABLE.
- FAVORISER L'APPRENTISSAGE PRATIQUE DES NOTIONS DE BASE DE DONNÉES, D'INTERFACES GRAPHIQUES ET DE PROGRAMMATION ORIENTÉE ÉVÉNEMENT.
- CE PROJET MET AINSI EN ÉVIDENCE LA COMPLÉMENTARITÉ ENTRE LA THÉORIE APPRISE EN COURS ET SA MISE EN ŒUVRE DANS UNE APPLICATION CONCRÈTE ET FONCTIONNELL

PRÉSENTATION DE L'APPLICATION

1-Les principales fonctionnalité

- **Gestion des films**

L'application permet à l'administrateur d'ajouter, de modifier et de supprimer des films. Pour chaque film, l'administrateur peut saisir des informations comme le titre, le genre, la durée et l'année de sortie. Cela permet de maintenir une base de données à jour des films disponibles à la projection.



- **Gestion des clients**

Il est possible d'ajouter de nouveaux clients en enregistrant leur nom, numéro de téléphone et adresse email. Ces informations facilitent la gestion des réservations et permettent de suivre l'historique des clients.

- **Gestion des réservations**

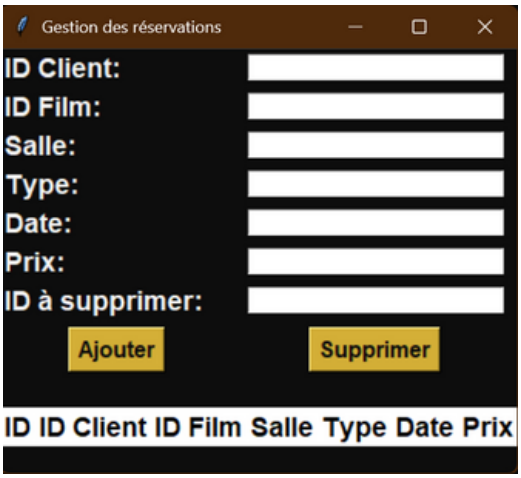
L'administrateur peut enregistrer des réservations en associant un client à un film, une salle, un type de projection (2D, 3D, etc.) et un prix. L'application permet également de supprimer des réservations et d'afficher toutes les réservations en cours.

2-Interfaces utilisateurs

L'application est équipée d'une interface graphique simple et facile à utiliser, développée avec la bibliothèque Tkinter. Elle est divisée en plusieurs fenêtres :

Fenêtre principale : Présente les différentes sections accessibles (Films, Clients, Réservations).

Fenêtres de gestion : Chaque fonction (films, clients, réservations) dispose de sa propre fenêtre pour ajouter, afficher ou supprimer des informations.



Tableaux : Les films, les clients et les réservations sont affichés dans des tableaux clairs, permettant à l'utilisateur de visualiser les données rapidement.

PRÉSENTATION DE CODE

L'application de gestion de cinéma est conçue pour être simple, efficace et intuitive, à la fois pour les administrateurs et les utilisateurs. Le code qui la soutient est structuré de manière modulaire et claire, permettant une gestion fluide des films, des clients et des réservations. Voici une présentation générale des parties principales du code.

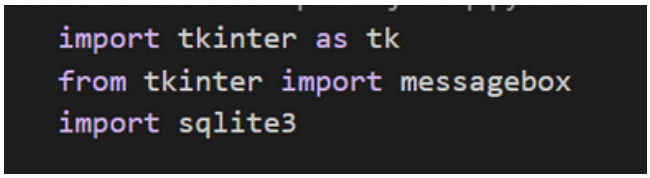
1. Bibliothèques et Dépendances

Le code utilise principalement deux bibliothèques :

Tkinter : Pour l'interface graphique.

SQLite : Pour la gestion des données (films, clients, réservations).

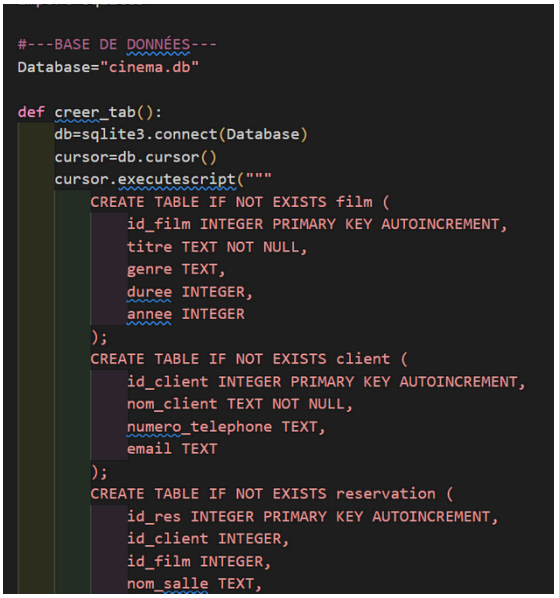
```
import tkinter as tk
from tkinter import messagebox
import sqlite3
```



CES BIBLIOTHÈQUES PERMETTENT DE CRÉER UNE INTERFACE GRAPHIQUE FLUIDE ET DE MANIPULER UNE BASE DE DONNÉES LÉGÈRE POUR STOCKER TOUTES LES INFORMATIONS NÉCESSAIRES.

2. INITIALISATION DE LA BASE DE DONNÉES

L'APPLICATION NÉCESSITE UNE BASE DE DONNÉES SQLITE POUR STOCKER LES INFORMATIONS DES FILMS, DES CLIENTS ET DES RÉSERVATIONS. LA FONCTION CI-DESSOUS MONTRE COMMENT LA BASE DE DONNÉES EST INITIALISÉE, AVEC LA CRÉATION DES TABLES NÉCESSAIRES.



EXEMPLE DE CODE :

```
DEF CREATE_DB():
    CONN =
SQLITE3.CONNECT('CINEMA.DB')
    CURSOR = CONN.CURSOR()
    CURSOR.EXECUTE('''
        CREATE TABLE IF NOT EXISTS
FILMS (
            ID INTEGER PRIMARY KEY,
            TITRE TEXT,
            GENRE TEXT,
            ANNEE INTEGER
        )
    ''')
    CONN.COMMIT()
    CONN.CLOSE()
```

EXPLICATION :

CETTE FONCTION CRÉE UNE TABLE FILMS DANS LA BASE DE DONNÉES POUR STOCKER LES INFORMATIONS DES FILMS. D'AUTRES TABLES SONT CRÉÉES DE MANIÈRE SIMILAIRE POUR LES CLIENTS ET LES RÉSERVATIONS.

3-INTERFACE UTILISATEUR (UI) AVEC TKINTER

L'interface utilisateur est construite à l'aide de Tkinter, une bibliothèque simple mais puissante pour créer des interfaces graphiques en Python. La fenêtre principale permet à l'administrateur de naviguer entre les différentes sections : films, clients et réservations.

```
def window_films():

    def ajouter_film():
        if not titre_var.get() or not genre_var.get() or not duree_var.get() or not annee_var.get():
            messagebox.showwarning("Attention", "Veuillez remplir tous les champs !")
            return
        inser_film(titre_var.get(),genre_var.get(),duree_var.get(),annee_var.get())
        afficher_films()
        messagebox.showinfo("Succès", "Film ajouté avec succès !")

    def supprimer_film():
        if not id_sup_var.get():
            messagebox.showwarning("Attention", "Veuillez entrer l'ID à supprimer !")
            return
        supp("film", "id_film", id_sup_var.get())
        afficher_films()

    tk.Button(fen, text="Ajouter", bg=BOUTON, fg=BOUTON_TEXTE, font=FONT_BTN, command=ajouter_film).grid(row=5, column=0, pady=5)
    tk.Button(fen, text="Supprimer", bg=BOUTON, fg=BOUTON_TEXTE, font=FONT_BTN, command=supprimer_film).grid(row=5, column=1, pady=5)

    afficher_films()
```

CETTE PARTIE DU CODE CRÉE UNE FENÊTRE PRINCIPALE AVEC DES BOUTONS ET DES FORMULAIRES POUR AJOUTER DES FILMS, CLIENTS ET RÉSERVATIONS. LE BOUTON "AJOUTER UN FILM" OUVRE UNE FENÊTRE SECONDAIRE PERMETTANT À L'ADMINISTRATEUR D'AJOUTER UN FILM.

```
tk.Button(fen, text="Ajouter", bg=BOUTON, fg=BOUTON_TEXTE, font=FONT_BTN, command=ajouter_film).grid(row=5, column=0, pady=5)
tk.Button(fen, text="Supprimer", bg=BOUTON, fg=BOUTON_TEXTE, font=FONT_BTN, command=supprimer_film).grid(row=5, column=1, pady=

afficher_films()
```

4-AJOUT D'UN FILM DANS LA BASE DE DONNÉES:

UNE FONCTIONNALITÉ CLÉ DE L'APPLICATION EST D'AJOUTER UN FILM DANS LA BASE DE DONNÉES. L'ADMINISTRATEUR PEUT SAISIR LES INFORMATIONS DU FILM (TITRE, GENRE, ANNÉE) VIA L'INTERFACE GRAPHIQUE. CES INFORMATIONS SONT ENSUITE ENREGISTRÉES DANS LA BASE DE DONNÉES.

NB:CE PRINCIPE S'APPLIQUE TOTALEMENT POUR LA TABLE "CLIENT" ET " RESERVATION".

5-AFFICHAGE DES DONNÉES (FILMS, CLIENTS, RÉSERVATIONS):

LES INFORMATIONS SUR LES FILMS, LES CLIENTS ET LES RÉSERVATIONS SONT AFFICHÉES DANS DES TABLEAUX À L'INTÉRIEUR DE L'APPLICATION. CELA PERMET AUX ADMINISTRATEURS DE VISUALISER RAPIDEMENT LES DONNÉES ENREGISTRÉES.

```python
def afficher_films():
    for widget in frame_affichage.winfo_children():
        widget.destroy()
    db = sqlite3.connect(Database)
    db.row_factory = sqlite3.Row
    cursor = db.execute("SELECT * FROM film")
    # En-tête
    tk.Label(frame_affichage, text="ID", bg=GRID_BG, fg=GRID_TEXT, font=FONT_LABEL).grid(row=0, column=0)
    tk.Label(frame_affichage, text="Titre", bg=GRID_BG, fg=GRID_TEXT, font=FONT_LABEL).grid(row=0, column=1)
    tk.Label(frame_affichage, text="Genre", bg=GRID_BG, fg=GRID_TEXT, font=FONT_LABEL).grid(row=0, column=2)
    tk.Label(frame_affichage, text="Durée", bg=GRID_BG, fg=GRID_TEXT, font=FONT_LABEL).grid(row=0, column=3)
    tk.Label(frame_affichage, text="Année", bg=GRID_BG, fg=GRID_TEXT, font=FONT_LABEL).grid(row=0, column=4)
    ligne = 1
    for row in cursor:
        tk.Label(frame_affichage, text=row['id_film'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=0)
        tk.Label(frame_affichage, text=row['titre'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=1)
        tk.Label(frame_affichage, text=row['genre'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=2)
        tk.Label(frame_affichage, text=row['duree'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=3)
        tk.Label(frame_affichage, text=row['annee'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=4)
        ligne+=1
    db.close()
```

```python
def afficher_clients():
    for widget in frame_affichage.winfo_children():
        widget.destroy()
    db = sqlite3.connect(Database)
    db.row_factory = sqlite3.Row
    cursor = db.execute("SELECT * FROM client")
    tk.Label(frame_affichage, text="ID", bg=GRID_BG, fg=GRID_TEXT, font=FONT_LABEL).grid(row=0, column=0)
    tk.Label(frame_affichage, text="Nom", bg=GRID_BG, fg=GRID_TEXT, font=FONT_LABEL).grid(row=0, column=1)
    tk.Label(frame_affichage, text="Téléphone", bg=GRID_BG, fg=GRID_TEXT, font=FONT_LABEL).grid(row=0, column=2)
    tk.Label(frame_affichage, text="Email", bg=GRID_BG, fg=GRID_TEXT, font=FONT_LABEL).grid(row=0, column=3)
    ligne = 1
    for row in cursor:
        tk.Label(frame_affichage, text=row['id_client'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=0)
        tk.Label(frame_affichage, text=row['nom_client'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=1)
        tk.Label(frame_affichage, text=row['numero_telephone'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=2)
        tk.Label(frame_affichage, text=row['email'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=3)
        ligne += 1
    db.close()
```

```python
def afficher_reservations():
    for widget in frame_affichage.winfo_children():
        widget.destroy()
    db = sqlite3.connect(Database)
    db.row_factory = sqlite3.Row
    cursor = db.execute("SELECT * FROM reservation")
    headers = ["ID", "ID Client", "ID Film", "Salle", "Type", "Date", "Prix"]
    for col, h in enumerate(headers):
        tk.Label(frame_affichage, text=h, bg=GRID_BG, fg=GRID_TEXT, font=FONT_LABEL).grid(row=0, column=col)
    ligne = 1
    for row in cursor:
        tk.Label(frame_affichage, text=row['id_res'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=0)
        tk.Label(frame_affichage, text=row['id_client'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=1)
        tk.Label(frame_affichage, text=row['id_film'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=2)
        tk.Label(frame_affichage, text=row['nom_salle'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=3)
        tk.Label(frame_affichage, text=row['type_de_projection'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=4)
        tk.Label(frame_affichage, text=row['date_res'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=5)
        tk.Label(frame_affichage, text=row['prix'], bg=GRID_BG, fg=GRID_TEXT).grid(row=ligne, column=6)
        ligne += 1
    db.close()
```

EXPLICATION :

CETTE FONCTION ENREGISTRE UNE RÉSERVATION EN ASSOCIANT UN CLIENT À UN FILM. LES INFORMATIONS NÉCESSAIRES SONT RÉCUPÉRÉES ET AJOUTÉES À LA TABLE RESERVATIONS.

CONCLUSION

Au terme de ce projet intitulé « Application de gestion de cinéma», nous avons pu mettre en œuvre les compétences acquises tout au long du module de Python avancé.

Ce travail nous a permis d'approfondir notre compréhension de la programmation orientée projet, tout en consolidant nos connaissances sur deux aspects essentiels :

- la création d'interfaces graphiques à l'aide du module Tkinter,
- et la gestion des bases de données relationnelles via SQLite3.

L'application développée répond aux besoins fondamentaux d'un système de gestion : l'ajout, l'affichage et la suppression des films, des clients et des réservations.

Elle se distingue par sa simplicité d'utilisation, sa palette de couleurs harmonieuse, et son interface intuitive. Ce projet a également renforcé notre capacité à travailler de manière structurée, à résoudre des problèmes techniques, et à produire un code clair et fonctionnel.
Nous considérons cette réalisation comme une base solide pour d'éventuelles améliorations futures, telles que l'ajout de fonctionnalités de mise à jour, de recherche avancée, ou encore de génération de rapports automatiques.

◆ Remerciements:
Nous tenons à exprimer notre profonde gratitude à notre enseignant pour son accompagnement constant, ses conseils pertinents et sa disponibilité tout au long de ce projet.
Ses explications claires et son encadrement bienveillant ont largement contribué à la réussite de ce travail et à l'enrichissement de nos compétences en développement d'applications Python.