ENSAM
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

COMPTE RENDU DU MINI-PROJET

POO PYHTON ADV

réalisé par:
EL MAAROUFI SOUKAINA
EL HAMOUDI WIAM

Encadré par:
HAIN MUSTAPHA

Introduction

Contexte du Projet

Dans le cadre de notre formation en programmation Python, nous avons développé une application de gestion des travaux pratiques. Ce projet vise à automatiser et simplifier la gestion des informations relatives aux professeurs, aux TP et aux étudiants dans un contexte académique.

Objectifs

Les objectifs principaux de ce projet sont :
Concevoir une application de gestion complète avec Python
Manipuler une base de données SQLite pour le stockage des informations
Créer une interface graphique intuitive avec Tkinter
Implémenter les opérations CRUD (Create, Read, Update, Delete)
Respecter les bonnes pratiques de programmation

Technologies Utilisées

Langage : Python 3.x
Base de données : SQLite3
Interface graphique : Tkinter
Gestion des fichiers : Module os pour les chemins relatifs

PRÉSENTATION DU PROJET

Description Générale

L'application "Gestion des TP" est un système permettant de gérer trois entités principales : les professeurs, les travaux pratiques et les étudiants. Elle offre une interface graphique conviviale permettant d'effectuer toutes les opérations nécessaires à la gestion quotidienne des TP.

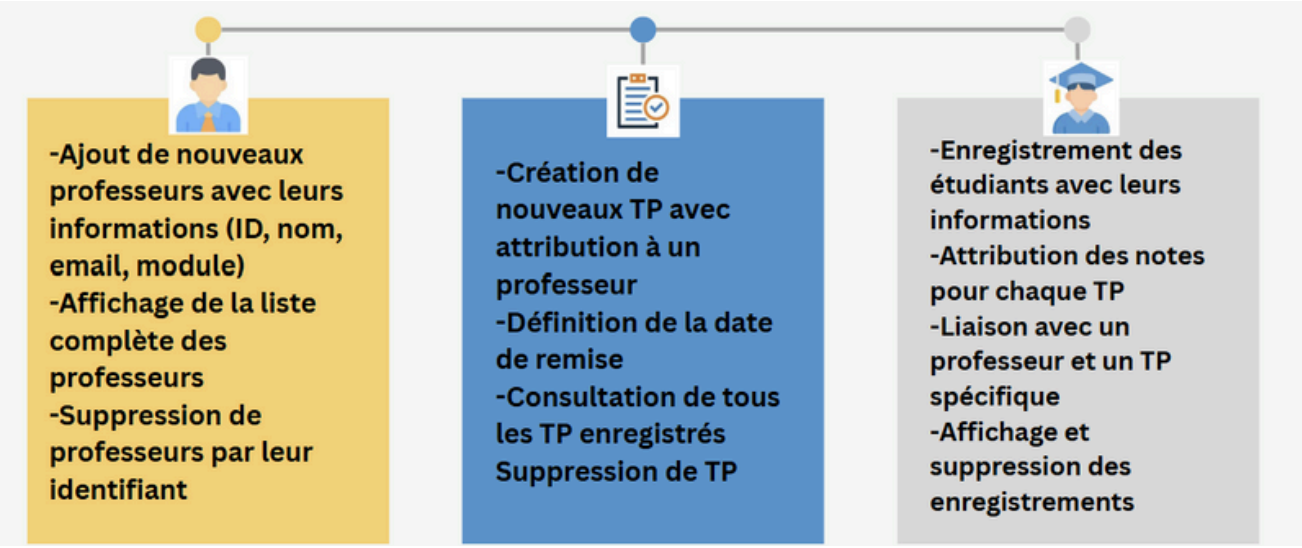
Fonctionnalités Principales

L'application propose les fonctionnalités suivantes :

Gestion des Professeurs

Gestion des TP

Gestion des Étudiants



BASE DE DONNÉES

Table Prof (Professeurs) :
teacher_id (INTEGER) : Identifiant unique du professeur
nom (TEXT) : Nom complet du professeur
email (TEXT) : Adresse email professionnelle
module (TEXT) : Module enseigné

Table TP (Travaux Pratiques) :
id (INTEGER) : Identifiant unique du TP
nom (TEXT) : Titre du TP
teacher_id (TEXT) : Référence au professeur responsable
module (TEXT) : Module concerné
Date_remise (TEXT) : Date limite de remise

Table Etudiant (Étudiants) :
id (INTEGER) : Identifiant unique de l'étudiant
nom (TEXT) : Nom complet de l'étudiant
teacher_id (TEXT) : Référence au professeur
Tp_id (INTEGER) : Référence au TP concerné
Note (INTEGER) : Note obtenue

Relations entre les Tables

Les relations établies sont :
-Un professeur peut être responsable de plusieurs TP)
-Un étudiant peut être évalué sur plusieurs TP
-Chaque TP est lié à un professeur spécifique

```
DB = sqlite3.connect(os.path.join(BASE_DIR, "gestion_des_TP.db"))

DB.execute("create table if not exists Prof (...)")
DB.execute("create table if not exists TP (...))")
DB.execute("create table if not exists Etudiant (...)")

DB.row_factory = sqlite3.Row
DB.commit()
```

La propriété row_factory permet d'accéder aux colonnes par leur nom.
create table if not exists: garantit que les tables sont créées uniquement si elles n'existent pas déjà.

gestion_des_TP.db. Ce fichier SQLite conserve toutes les informations même après la fermeture de l'application, permettant ainsi de retrouver l'ensemble des professeurs, TP et étudiants lors de la prochaine utilisation. Cette persistance des données est essentielle pour une application de gestion, éliminant le besoin de ressaisir les informations à chaque démarrage.

INTERFACES GRAPHIQUES

Fenêtre Principale

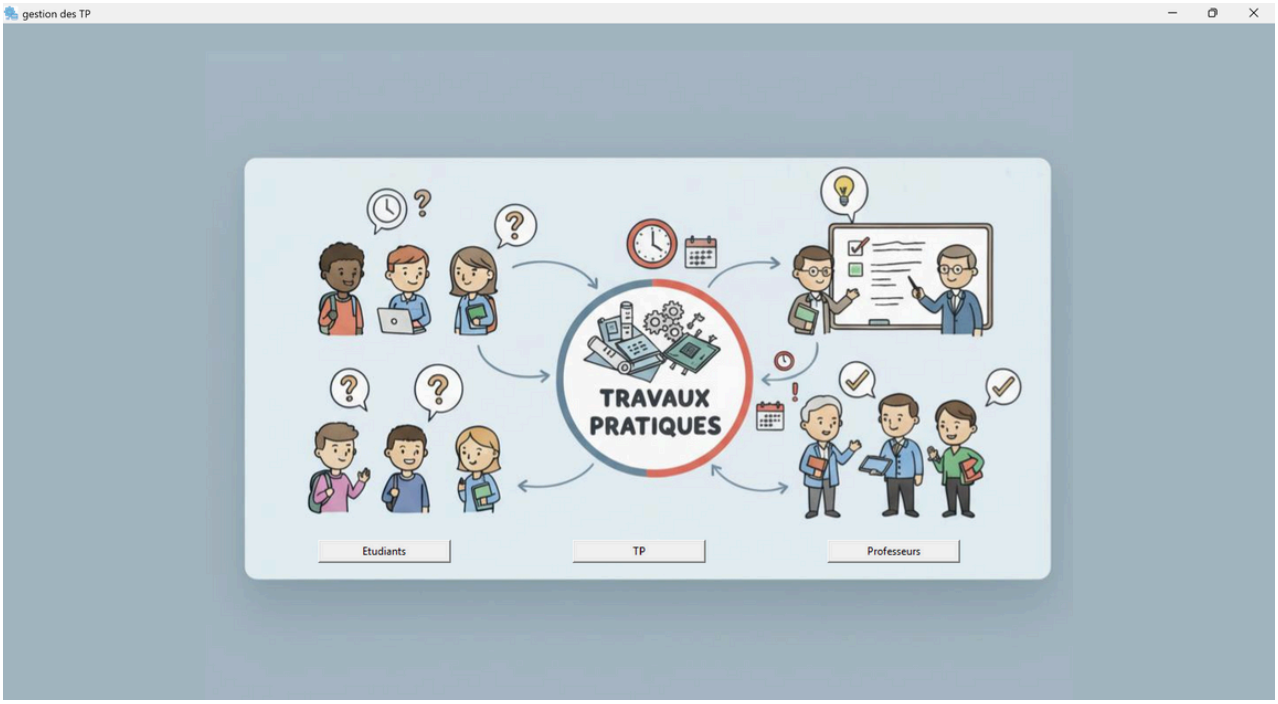
La fenêtre principale constitue le point d'entrée de l'application. Elle présente trois boutons principaux permettant d'accéder aux différents modules de gestion :

Caractéristiques :

Dimensions : 1000x600 pixels
Couleur de fond : #a3b7c2 (bleu clair)
Image de fond personnalisée
Trois boutons centrés : Étudiants, TP, Professeurs

Positionnement des boutons :

```
btn_prof.place(relx=0.7, rely=0.72, anchor=CENTER)
btn_TP.place(relx=0.5, rely=0.72, anchor=CENTER)
btn_Etudiant.place(relx=0.3, rely=0.72, anchor=CENTER)
```



Interface Professeurs

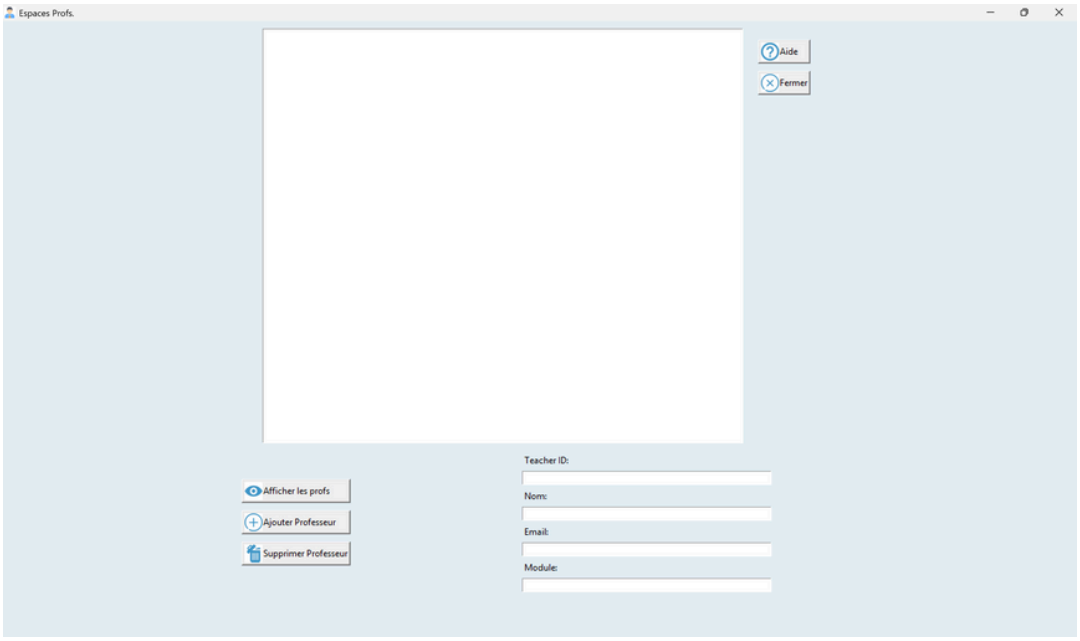
Cette fenêtre permet la gestion complète des professeurs :

Éléments visuels :

-Zone de texte (Text widget) pour l'affichage tabulaire des données
-Champs de saisie pour : Teacher ID, Nom, Email, Module
-Boutons d'action avec icônes : Afficher, Ajouter, Supprimer
-Boutons utilitaires : Aide, Fermer

Disposition :

-Frame supérieur : affichage des données
-Frame inférieur : formulaire de saisie et boutons d'action
-Organisation en colonnes pour une meilleure lisibilité

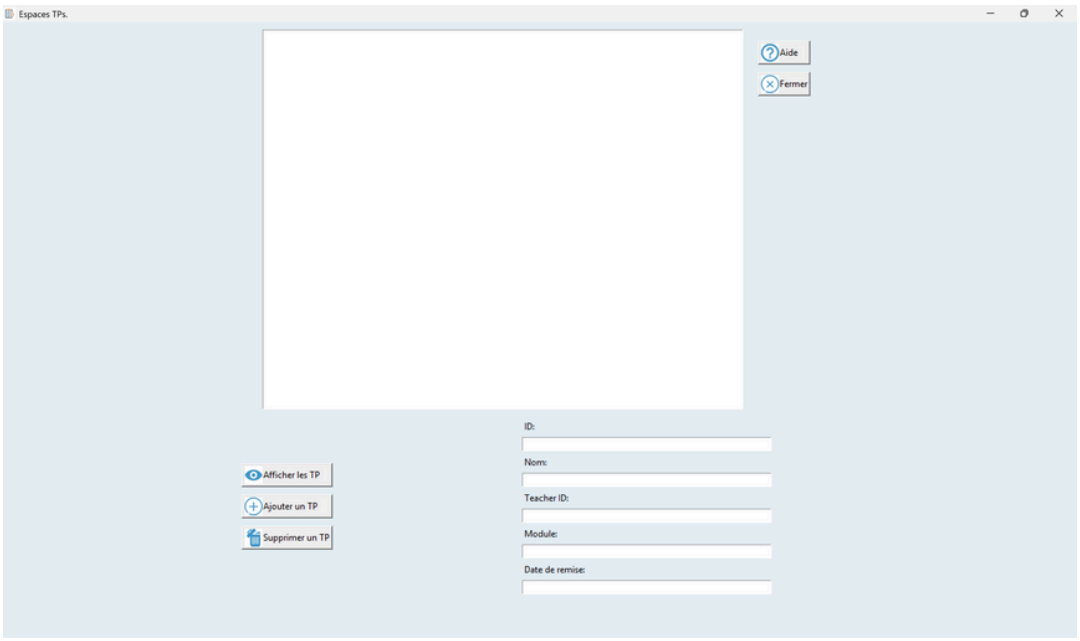


Interface TP

Interface dédiée à la gestion des travaux pratiques :

Champs de saisie :

-ID du TP
-Nom du TP
-Teacher ID (professeur responsable)
-Module
-Date de remise
-Affichage : Format tabulaire avec colonnes alignées pour une lecture facilitée



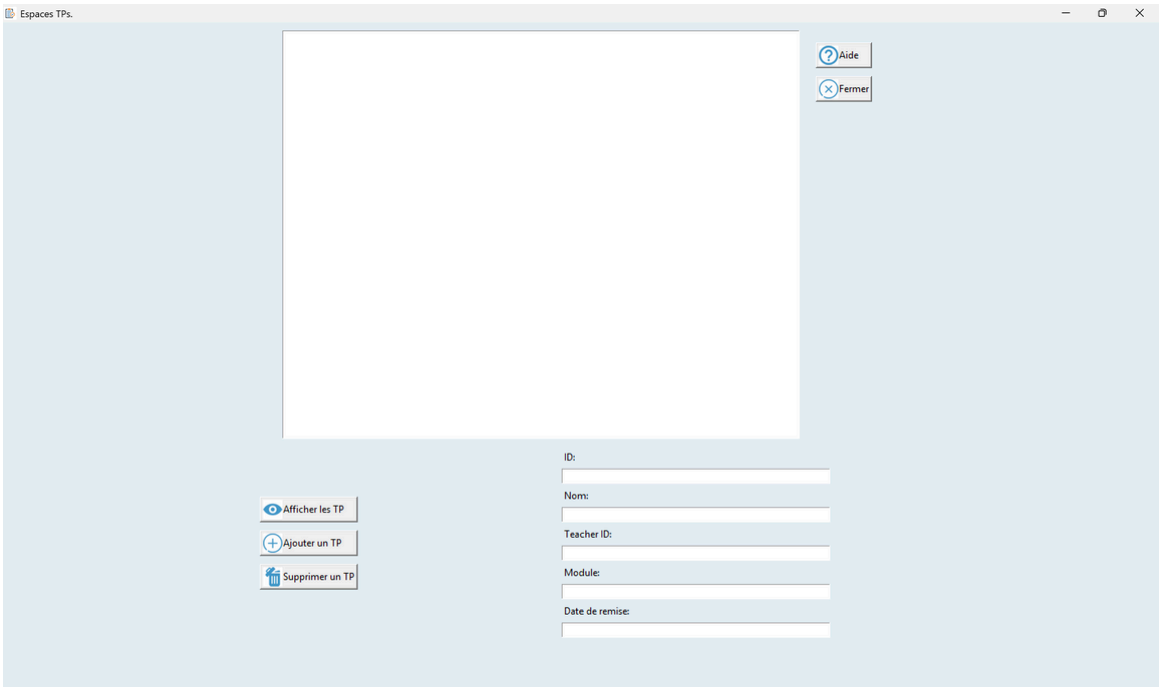
Interface Étudiants

Fenêtre de gestion des étudiants et de leurs notes :

Informations gérées :

-ID étudiant
-Nom complet
-Teacher ID
-TP ID
-Note

Particularités : La suppression s'effectue par TP_id pour permettre de retirer tous les étudiants liés à un TP spécifique



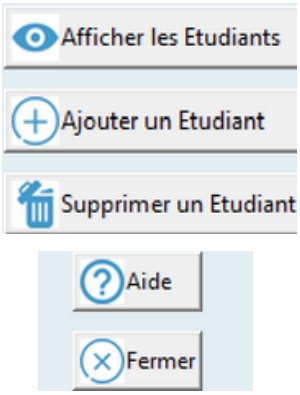
Système d'Icônes

L'application utilise un système de chargement dynamique des icônes :

```python
icons = {}

def load_icons():
    """Charge toutes les icônes nécessaires pour les boutons"""
    icon_names = {
        'ajouter': 'ajouter.png',
        'afficher': 'afficher.png',
        'supprimer': 'supprimer.png',
        'aide': 'aide.png',
        'fermer': 'fermer.png',
        'main': 'gestion_des_tps.png',
        'prof': 'prof.png',
        'tp': 'Tps.png',
        'etudiant': 'etudiant.png'
    }

    for name, filename in icon_names.items():
        try:
            icon_path = os.path.join(BASE_DIR, "icons_resized", filename)
            icons[name] = PhotoImage(file=icon_path)
        except:
            icons[name] = None
            print(f"Impossible de charger l'icône: {name}")
```



Opération d'Affichage (READ)

L'affichage présente les données sous forme tabulaire :

```
def Afficher_Prof():
    text_box.delete("1.0", END)
    cursor = DB.execute("select * from Prof")
    text_box.insert(END, "teacher_id   | Nom          | Email              | Module\n")
    text_box.insert(END, "-"*60 + "\n")
```

Avantages :
Format lisible et aligné
En-têtes de colonnes clairs
Séparateur visuel

```
teacher_id   | Nom          | Email              | Module
------------------------------------------------------------
17178        |salim         |salim@gmail.com     |MDP
779900       |yaakoubi      |yaakoubi@gmail.com  |thermodynamique
```

Opération de Suppression (DELETE)

La suppression s'effectue par identifiant :

```
def remove_prof():
    DB.execute("delete from Prof where teacher_id=?",
               (int(teacher_id_var.get()),))
    DB.commit()
```

Gestion des Exceptions

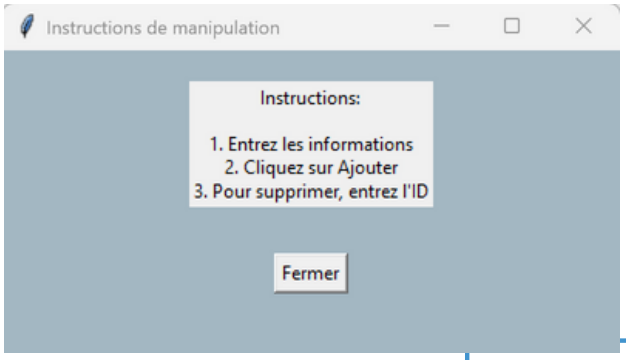
```
try:
    bg_image = PhotoImage(file=os.path.join(BASE_DIR, "PNG_POO", "image.png"
    bg_label = Label(main, image=bg_image, bg='#a3b7c2')
    bg_label.place(x=0, y=0, relwidth=1, relheight=1)
except:
    pass
```

Cela garantit que l'application fonctionne même si certains fichiers sont absents.

Système d'Aide

Chaque fenêtre dispose d'un bouton d'aide affichant des instructions contextuelles :

```
def instructions():
    aide = Toplevel(window1)
    aide.title("Instructions de manipulation")
    aide.geometry("400x300")
    aide.configure(bg="#a3b7c2")
    Label(aide, text="Instructions:\n\n1. Entrez les informations\n2.
    Button(aide, text="Fermer",  command=aide.destroy).pack(pady=10)
```



CODE SOURCE ESSENTIEL

Initialisation de l'Application

```
import sqlite3
from tkinter import *
import os

BASE_DIR = os.path.dirname(os.path.abspath(__file__))

main = Tk()
main.title("gestion des TP")
main.geometry("1000x600")
main.configure(bg="#a3b7c2")
```

Structure d'une Fenêtre Secondaire

```
def Prof_window():
    window1 = Toplevel(main)
    window1.title("Espaces Profs.")
    window1.geometry("1000x600")

    # Création des frames
    main_frame = Frame(window1, bg="#e3eef4")
    main_frame.pack(fill=BOTH, expand=True)

    # Zone d'affichage
    text_box = Text(top_frame, height=15, width=80)
    text_box.pack(side=LEFT, fill=BOTH, expand=True)
```

Points de Validation

✓ Base de données créée automatiquement
✓ Tables correctement structurées
✓ Opérations d'ajout fonctionnelles
✓ Affichage formaté et lisible
✓ Suppression sécurisée
✓ Navigation fluide entre les fenêtres
✓ Gestion des erreurs

DIFFICULTÉS RENCONTRÉES

Au cours du développement de cette application, plusieurs difficultés ont été rencontrées :
Gestionnaire de positionnement Grid : La plus grande difficulté a été l'utilisation du gestionnaire grid() qui ne permettait pas de modéliser l'interface selon la vision souhaitée. Les contraintes de lignes et colonnes rigides rendaient difficile l'obtention d'une disposition flexible et esthétique. C'est pourquoi nous avons opté pour une combinaison de pack() et place() qui offre plus de liberté dans le positionnement des widgets.
Gestion des chemins relatifs : Les chemins absolus rendaient l'application non portable. Solution : utilisation du module os avec BASE_DIR pour calculer les chemins relatifs.
Chargement des icônes : Certaines icônes ne se chargeaient pas, provoquant des erreurs. Solution : implémentation d'un système de gestion d'exceptions avec blocs try-except.

Structure du Projet

```
Projet_Gestion_TP/
│
├── gestion_tp.py           # Code source principal
├── gestion_des_TP.db       # Base de données SQLite (générée
automatiquement)
│
├── icons_resized/          # Dossier des icônes pour l'interface
│   ├── ajouter.png
│   ├── afficher.png
│   ├── supprimer.png
│   ├── aide.png
│   ├── fermer.png
│   ├── prof.png
│   ├── Tps.png
│   ├── etudiant.png
│   └── gestion_des_tps.png
│
└── PNG_POO/                # Ressources visuelles
    └── image.png           # Image de fond
```

Prérequis d'Installation

-Logiciels nécessaires :
-Python 3.7 ou supérieur
-Tkinter (généralement inclus avec Python)
-SQLite3 (inclus avec Python)

CONCLUSION

Ce projet "Gestion des TP" vise à développer une application complète permettant de gérer les professeurs, les travaux pratiques et les étudiants dans un contexte académique. En intégrant Python, SQLite et Tkinter, nous avons pu créer une solution fonctionnelle répondant aux besoins de gestion quotidienne des TP.

Nous remercions notre professeur de Python Avancé pour ses explications claires durant les cours, qui nous ont permis de réaliser ce projet. Ses cours nous ont fourni les notions de base nécessaires pour comprendre et appliquer les concepts abordés. Ce projet nous a également permis de mettre en pratique ces notions et de renforcer notre compréhension du langage Python.

Ce projet a été une expérience enrichissante qui nous a permis de mettre en pratique les concepts fondamentaux de la programmation Python. Les objectifs fixés ont été pleinement atteints : l'application offre une interface intuitive permettant de gérer efficacement les professeurs, les TP et les étudiants, avec une base de données relationnelle SQLite assurant la persistance des données. L'implémentation des opérations CRUD (Create, Read, Delete) fonctionne parfaitement, et l'interface facilite l'utilisation quotidienne du système.

Au niveau des compétences développées, ce projet nous a permis de maîtriser la manipulation de bases de données relationnelles avec SQLite, la conception d'interfaces graphiques avec Tkinter, l'organisation de code Python en fonctions modulaires, la gestion des erreurs et exceptions, ainsi que le travail avec les systèmes de fichiers et chemins relatifs. L'application développée, bien que simple dans sa conception, démontre une compréhension approfondie de l'intégration entre base de données et interface graphique, constituant ainsi une base solide pour des développements futurs plus ambitieux dans le domaine de la gestion d'applications.