RAPPORT DE PROJET PYTHON AVANCE

2025-2026

Encadre par:
Prof. Hain

Réalisée par:
Ziyad Ouarrad
Karim Elghmari

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

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

I. Introduction :

Le projet SGBDR-App est une application de gestion de bases de données relationnelles (Système de Gestion de Base de Données Relationnelle - SGBDR) développée en Python. Elle fournit une interface graphique conviviale, conçue avec Tkinter, permettant aux utilisateurs de créer, manipuler et partager des bases de données SQLite sans connaissance préalable du SQL. L'application inclut un système de gestion des utilisateurs, de permissions et de bases de données, tout en offrant la possibilité d'exporter les tables au format CSV.

Vous pouvez trouver le projet sur le lien GITHUB suivant :

https://github.com/ziyadouarrad-etu/SGBDR-App.git

II. Caractéristiques :

🔐 Gestion des Utilisateurs

- Système d'enregistrement et d'authentification des utilisateurs
- Accès sécurisé par utilisateur
- La table des permissions contrôle quels utilisateurs peuvent accéder à quelles bases de données
- Compte administrateur intégré(As a Back Door for testing) (mot de passe par défaut : test)

💾 Gestion des Bases de Données

- Création de nouvelles bases SQLite
- Gestion de plusieurs bases de données par utilisateur
- Partage des bases de données avec d'autres utilisateurs enregistrés
- Suppression des bases de données

📊 Opérations sur les Tables

- Création de tables avec des colonnes personnalisées et des types de données (TEXT, INTEGER, REAL, BLOB)

- Visualisation et navigation dans les tables
- Insertion de nouveaux enregistrements
- Suppression de certains enregistrements ou de tables entières
- Export des tables au format .csv pour une utilisation externe

🧩 Points Techniques Clés

- Interface graphique développée avec Tkinter et ttk
- Persistance des données via SQLite
- Mise en place automatique de user_management.db pour stocker les utilisateurs et les permissions
- Structure modulaire avec une classe DB gérant les opérations SQL de bas niveau
- Léger et portable — aucun serveur externe requis

III. Explication du code :

Pour le développement du projet, on a utilisé plusieurs classes :

DB
Properties : conn: The SQLite connection object (sqlite3.connect(...)) c: The cursor object for executing SQL commands (self.conn.cursor())
Methods: __init__(self, name) • Initializes the database connection with the given database name. create_table(self, table, cols) • Creates a table with a dictionary of columns and their data types. insert(self, table, data) • Inserts a row into the table. data is a dictionary of {column: value}. fetch_all(self, table) • Returns all rows from the specified table. get_columns(self, table) • Returns a list of column names for the table. get_tables(self) • Returns a list of all table names in the database. drop_table(self, table) • Deletes the table from the database. export_csv(self, table) • Exports the table data to a CSV file named <table>.csv. close(self) • Closes the database connection.

APP
Properties : root: Tkinter root window (tk.Tk()) user: Currently logged-in username (None before login) current_db: Name of the database currently opened
Methods: **Authentication** • login_screen(self): Displays the login GUI. • register_screen(self): Displays the registration GUI. • grant(self, db_name, username): Grants access to a database. • revoke(self, db_name, username): Revokes access to a database. **Database Management** • db_screen(self): Shows the dashboard with databases the user can access. • create_db(self): Prompts to create a new database. • access_screen(self, db_name): Screen for managing database sharing. • open_db(self, name): Opens a database and displays its tables. • delete_db(self): Deletes the current database and its permissions. **Table Management** • table_list_screen(self): Displays all tables of the current database. • create_table(self): Prompts to create a new table. • table_create_screen(self, table_name): GUI to create table columns. • table_screen(self, table): Displays records of a table. • add_record(self, table, cols): GUI to add a record to a table. • delete_table(self, table): Deletes a table from the database. • export(self, table): Exports a table to CSV. • delete_record(self, tree, table, cols): Deletes selected records from a table.

On a aussi utilisé plusieurs fonctions :

1. init_user_db()

- **But :** Initialise la base de données des utilisateurs (user_management.db) si elle n'existe pas.
- **Actions :**
  - Crée la table users (username, password)
  - Crée la table permissions (db_name, username, created_by)
  - Insère un compte administrateur par défaut : username = admin, password = test
- **Retour :** None

---

2. login(username, password)

- **But :** Vérifie les identifiants d'un utilisateur.
- **Entrées :**
  - username (str)
  - password (str)
- **Retour :** True si les identifiants sont corrects, False sinon

---

3. create_user(username, password)

- **But :** Crée un nouvel utilisateur dans la table users.
- **Retour :** True si l'utilisateur a été créé, False si le nom d'utilisateur existe déjà

---

4. get_user_dbs(username)

- **But :** Retourne toutes les bases de données auxquelles l'utilisateur a accès.
- **Retour :** Liste de dictionnaires :

[{"name": nom_base, "creator": createur}, ...]

---

5. add_permission(db_name, username, created_by)

- **But :** Donne l'accès à une base de données à un utilisateur.
- **Retour :** True si la permission est ajoutée, False si elle existe déjà

---

6. get_all_users()

- **But :** Récupère la liste de tous les utilisateurs enregistrés.
- **Retour :** Liste de chaînes de caractères (list[str])

---

7. get_db_users(db_name)

- **But :** Récupère la liste des utilisateurs ayant accès à une base spécifique.
- **Retour :** Liste de chaînes de caractères (list[str])

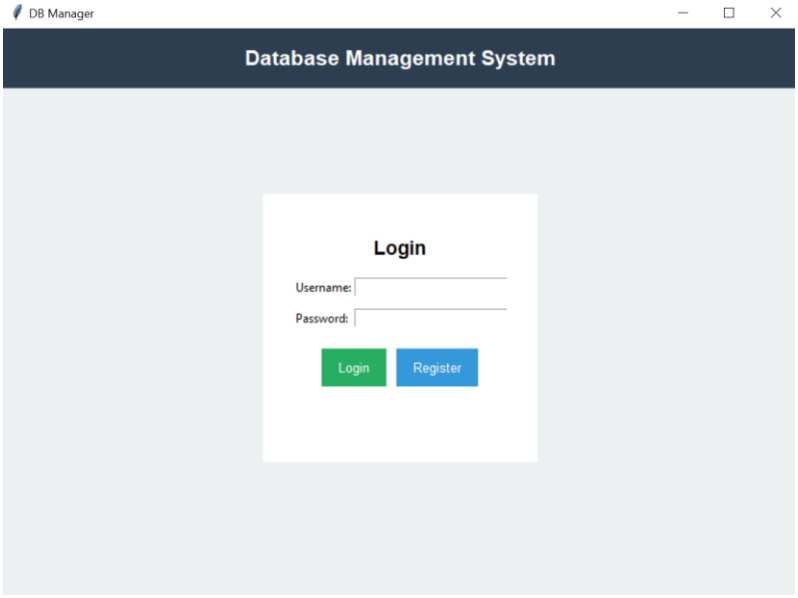
---

8. revoke_permission(db_name, username)

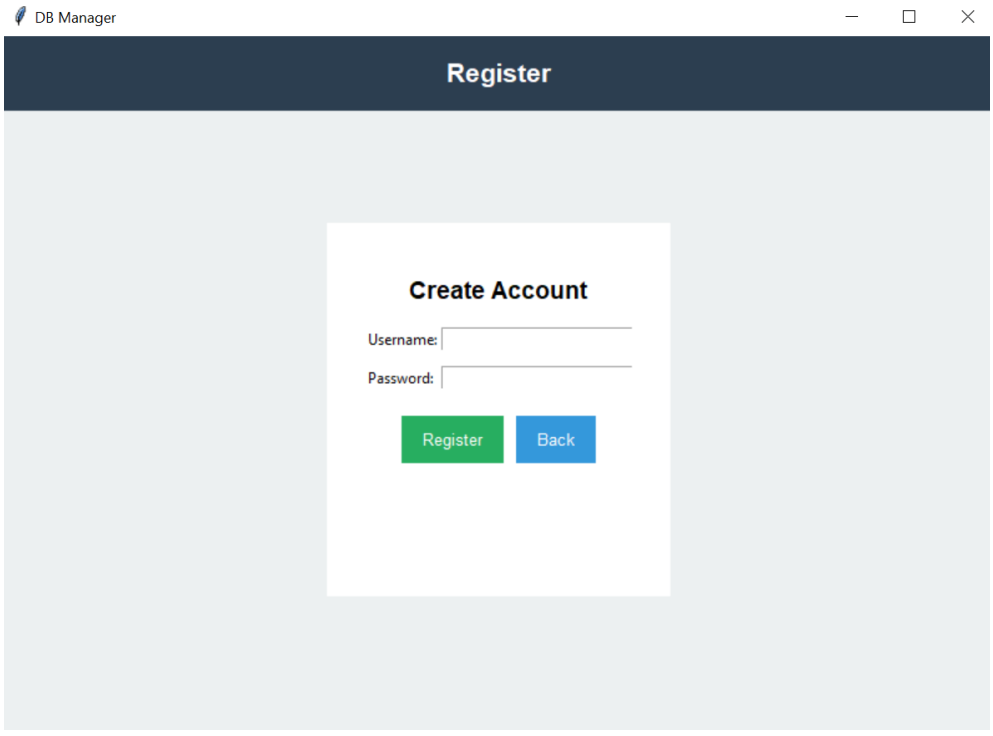
- **But :** Supprime l'accès d'un utilisateur à une base de données.
- **Retour :** None

IV. Démonstration et tour du Projet :

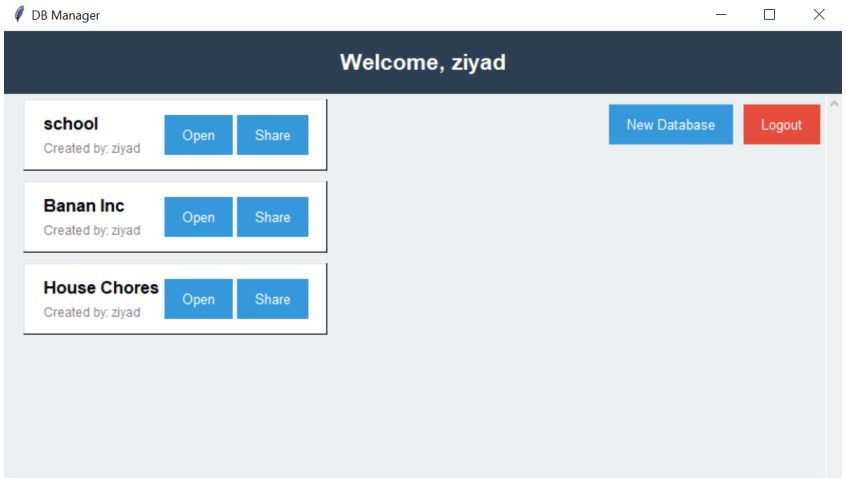
Quand le programmes est lancée, la page initiale tkinter qui s'affiche c'est la page de log in :



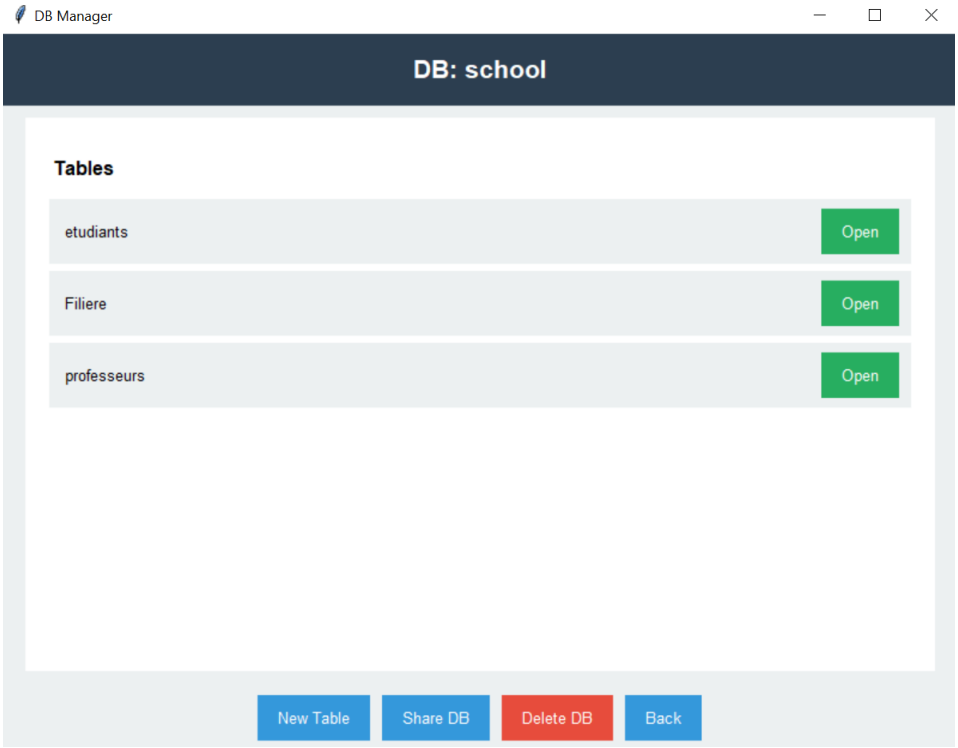
Si vous n'avez pas un compte vous pouvez en créer un en cliquant sur le bouton Register qui vous menera sur une autre page :



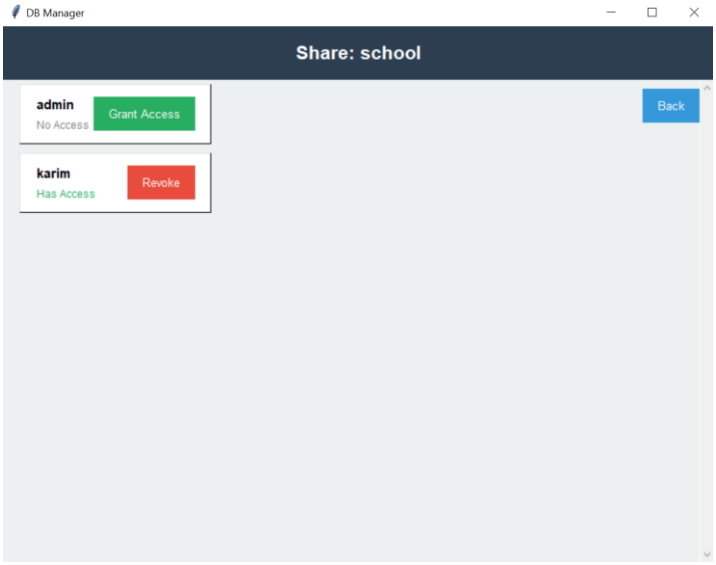
Quand vous vous connectez, vous allez trouver tous les Bases de Données que vous en avez accès (soit créé par vous-même ou une autre personne vous a donné accès à leur BD). Par exemple Ziyad a accès aux bases de donnes « school », « Banana Inc » and « House Chores », comme il peut aussi créer une nouvelle base de donne :



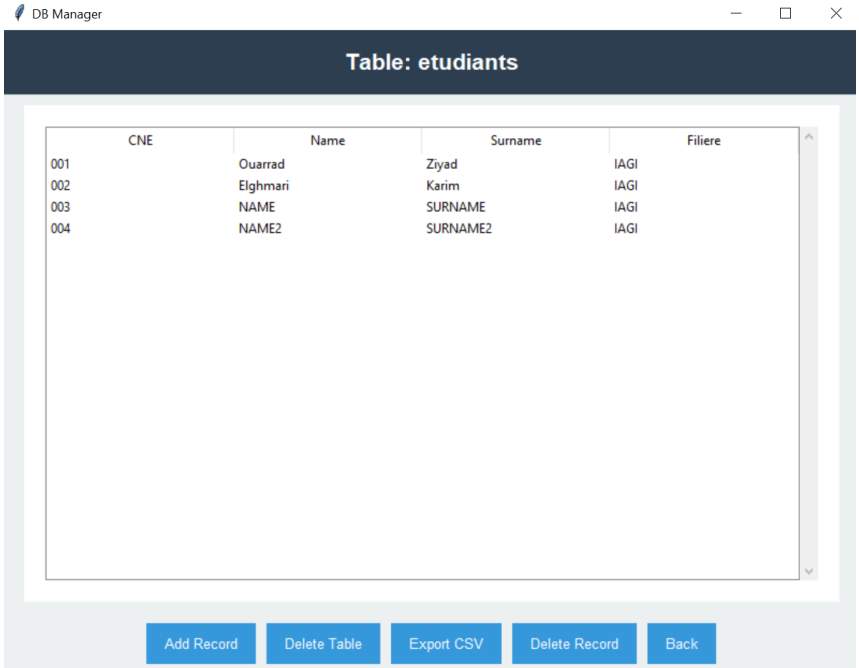
Lorsque vous ouvrez une base de donne, vous aurez acces a toutes les tables dedans (La BD school de Ziyad contient les tables « etudiants », « Filiere » et « professeurs »), et bien sûr il y'a l'option de créer une nouvelle table :



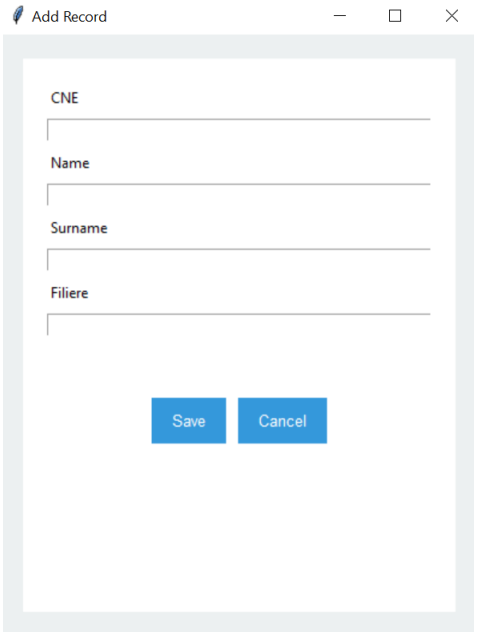
Avant d'ouvrir la table pour la sa gestion, jetons un œil sur l'option de « Share DB » qui vous permet d'accorder l'accès à votre DB à un autre utilisateur. Ici Karim a aussi accès à la BD « school ».



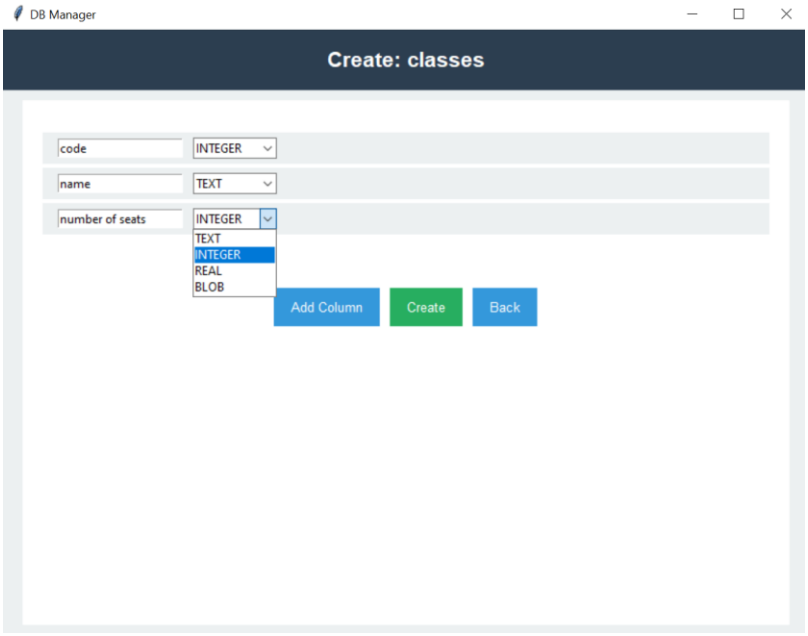
D'une autre part, si on accède à un tableau, on oura une visuelle sur tous les enregistrements dans ce dernier, et aussi plusieurs options de gestions comme l'ajout d'un élément, la suppression d'un ou plusieurs éléments sélectionnés, l'exportation de la table comme un fichier CSV ou même la suppression de le tableau entiers.



Et voici comment on peut ajouter un élément a un tableau, on ajoute juste les observations dans les cases concernant chaque colonne.



Dernièrement, jetons un œil sur l'ajout d'une nouvelle table à une BD : on fait entrer les noms des colonnes et on sélectionne le type, puis on click sur ajouter colonne pour la sauvegarder (on peut ajouter autant qu'on veut !)



V. Conclusion :

Le projet **SGBDR-App** illustre la conception et le développement d'un système de gestion de bases de données relationnelles avec interface graphique, entièrement réalisé en Python avec Tkinter et SQLite. Ce projet a permis de mettre en pratique des concepts fondamentaux des systèmes de gestion de bases de données, tels que la création de tables, la gestion des utilisateurs, les permissions d'accès et l'exportation de données.

L'application offre une interface conviviale qui facilite la manipulation des bases de données, même pour des utilisateurs n'ayant pas de connaissances avancées en SQL. Grâce à la modularité du code et à la classe DB, la manipulation des bases de données est centralisée et sécurisée, tandis que la classe App permet de gérer l'ensemble des interactions utilisateurs et la navigation dans l'interface.

Ce projet constitue également un excellent support pour comprendre la gestion multi-utilisateur, la sécurité des accès et les bonnes pratiques de structuration d'une application Python. Enfin, il offre une base solide pour de futures améliorations, telles que l'ajout de l'édition des enregistrements, le hachage des mots de passe, ou encore l'intégration d'une interface graphique plus moderne avec PyQt ou d'autres frameworks.