RAPPORT POO

Réalisé par:
EL HAMDI Khawla
EL MAGANA Fatima-Zahra
BITANE Malak

Encadré par :
Mr Mustapha Hain

Concept d'application

Explication du concept:

Icon du Logo

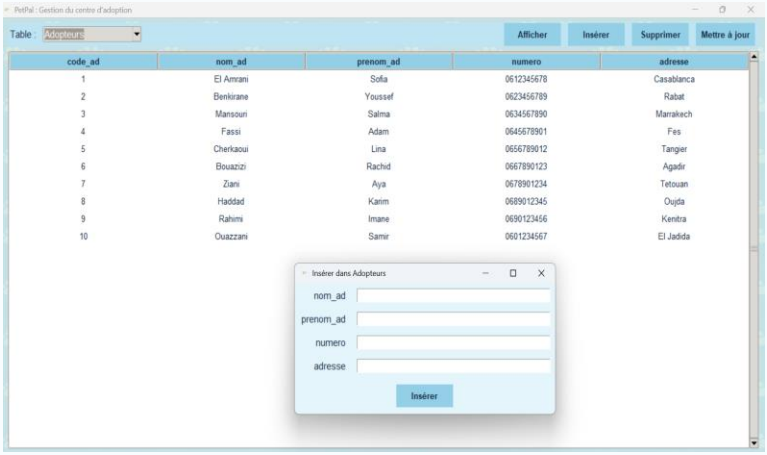
L'interface graphique de notre projet **PetPal** a pour objectif de rendre la gestion d'un centre d'adoption d'animaux simple, intuitive et interactive. Développée avec la bibliothèque Tkinter et ses widgets modernes ttk, elle permet à l'utilisateur d'interagir facilement avec la base de données du centre à travers une seule fenêtre ergonomique.Cette interface offre plusieurs fonctionnalités essentielles : **afficher, insérer, modifier et supprimer** les informations des trois tables principales — **Animaux**, **Adopteurs** et **Adoptions**. Les données sont présentées sous forme de tableau dynamique (Treeview), qui se met automatiquement à jour selon la table choisie. Des boutons clairs et bien disposés permettent de réaliser les différentes opérations, tandis que des fenêtres secondaires s'ouvrent pour l'ajout ou la mise à jour des enregistrements.

Sur le plan visuel, l'application adopte un design moderne et attractif, avec un fond d'écran redimensionnable, des couleurs harmonieuses et un logo représentatif du centre. Chaque action de l'utilisateur déclenche un message d'information ou d'erreur, garantissant une utilisation fluide et sécurisée.

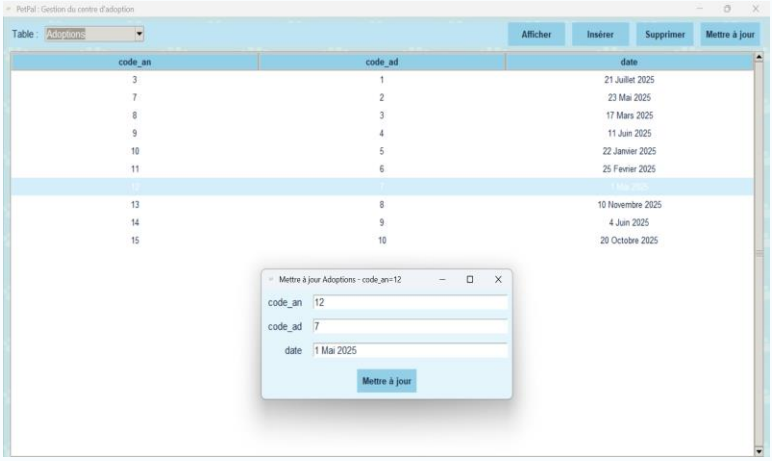
Ainsi, l'interface de **PetPal** relie efficacement la base de données SQLite au monde visuel, permettant une gestion complète et agréable du centre d'adoption tout en illustrant concrètement l'application des principes de la programmation orientée objet et événementielle.

Interface graphique:

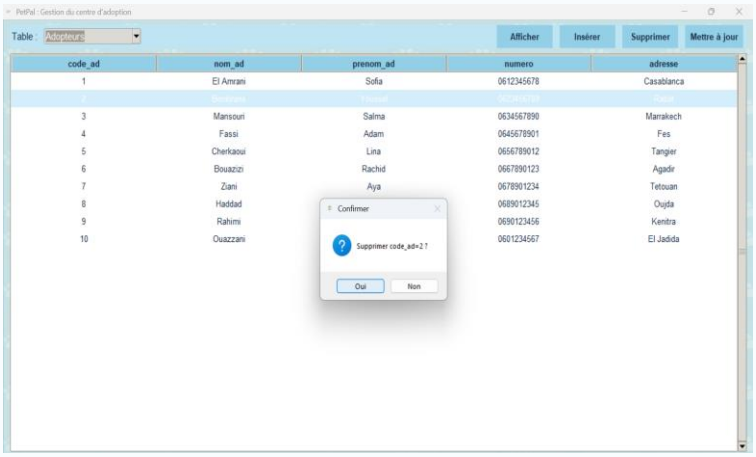
- Interface d'insertion permettant d'ajouter de nouveaux enregistrements dans la table sélectionnée.

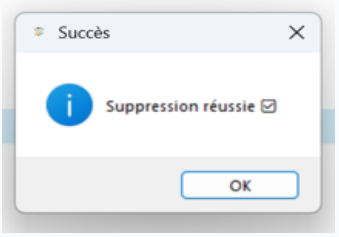


- Interface de mise à jour permettant de modifier les enregistrements existants dans la table sélectionnée.



- Interface de suppression permettant de supprimer définitivement l'enregistrement sélectionné sur la table.





Base de Données

Structure de la base de données:

Pour la mise en place d'une base de données relationnelle destinée à gérer les informations de ce centre d'adoption d'animaux ; nous avons utilisé le système de gestion de base de données SQLite3 intégré dans python. La base de données est stockée dans un fichier local, ce qui permet une portabilité facile du projet.
La base de données contient trois tables permettant de représenter les différents éléments du système d'adoption crée par :

Animaux : contient les informations sur les animaux du centre d'adoption :

- **Code_an** : identifiant unique de l'animal (clé primaire)
- **Nom_an** : nom de l'animal
- **Race** : race de l'animal
- **Statut** : « adopté » ou « non adopté »

Adopteurs : représente les personnes ayant adopté un animal du centre :

- Code_ad : identifiant unique de l'adopteur (clé primaire)
- **Nom_ad** : nom de l'adopteur
- **Prenom_ad** : prenom de l'adopteur
- **Numero** : numero de telephone de l'adopteur
- **Adresse** : adresse de l'adopteur

Adoptions : table reliant un adopteur avec son animal adopté :

- **Code_an** : identifiant de l'animal (clé étrangère)
- **Code_ad** : identifiant de l'adopteur (clé étrangère)
- **Date** : date d'adoption

Fonctions

Bibliothèques :

Dans ce projet, on a utilisé trois bibliothèques qui permettent d'assurer la **liaison entre l'interface graphique** et **la base de données**, et de rendre l'application à la fois fonctionnelle et simple d'utilisation :

- **tkinter** : Elle offre différents widgets (boutons, champs de saisie, tableaux, etc.) qui facilitent l'interaction de l'utilisateur avec l'application.
  Le module **ttk** apporte des éléments visuels plus modernes et stylés, utilisés notamment pour le tableau (Treeview).
- **messagebox** (sous-module de tkinter) : Il permet d'afficher des messages d'information, de confirmation ou d'erreur (par exemple lors d'une insertion ou suppression).
- **sqlite3** : Elle permet la connexion et la manipulation de la base de données SQLite. Elle est utilisée pour exécuter des requêtes SQL telles que SELECT, INSERT, UPDATE, ou DELETE.
- **PIL :** Elle permet de manipuler et de traiter des images dans l'application. On peut l'utiliser pour ouvrir, redimensionner, recadrer, convertir ou enregistrer des images dans différents formats.

Fonctions principales :

Dans ce projet, on a créé plusieurs fonctions qui permettent d'effectuer les opérations essentielles de gestion du centre d'adoption (affichage, ajout, suppression et mise à jour) (= modèle CRUD).

- **connect_db()** :
  Elle établit la connexion avec la base de données SQLite à partir de son chemin. Elle est utilisée dans toutes les autres fonctions pour assurer l'accès aux tables.
- **get_table_columns(table)** :
  Cette fonction récupère la liste des colonnes d'une table donnée. Elle sert principalement à configurer automatiquement l'affichage dans le tableau (Treeview).
- **show_data(table, tree)** :
  Cette fonction affiche le contenu d'une table dans l'interface. Elle exécute une requête SELECT *, récupère les données et les affiche dans le widget Treeview. Avant l'affichage, elle vide le tableau pour éviter les doublons.

- **insert_data_dynamic(table, entries_values)** :
  Cette fonction permet d'insérer de nouvelles données dans une table. Les champs et leurs valeurs sont récupérés directement depuis l'interface.
  De plus, si l'insertion concerne la table **Adoptions**, l'état de l'animal est automatiquement mis à jour en "Adopté".
- **delete_row(table, pk_col, pk_value)** :
  Cette fonction supprime une ligne sélectionnée dans une table. La suppression se fait en utilisant la clé primaire (pk_col) afin d'éviter les erreurs.
- **update_row(table, pk_col, pk_value, updates)** :
  Cette fonction permet de modifier des informations existantes dans la base. Elle met à jour uniquement les champs choisis par l'utilisateur puis applique la modification à l'enregistrement correspondant.

Interface graphique

Explication du style de l'interface :

Dans cette partie du projet PetPal, nous avons configuré le style visuel et l'apparence générale de l'interface graphique. L'objectif est de rendre l'application esthétique, lisible et agréable à utiliser.

- Un fond d'écran est appliqué, et il se redimensionne automatiquement lorsque l'utilisateur ajuste la taille de la fenêtre.
- Les couleurs principales ont été choisies pour harmoniser l'interface : un bleu foncé pour les textes et titres (main_color), un bleu clair pour les boutons (accent Blue), un fond lumineux (light_bg) et du blanc pour les tableaux.
- Les widgets comme les boutons (TButton) et les tableaux (Treeview) ont été stylés

Organisation de l'interface et affichage des données :

Le top frame sert de barre supérieure et contient un label et un menu déroulant (Combobox) permettant à l'utilisateur de choisir la table à afficher parmi Adopteurs, Animaux et Adoptions. Il inclut également un espace réservé aux boutons d'action placés à droite pour un accès rapide et clair.
Le tree frame occupe la majeure partie de la fenêtre et contient un tableau (Treeview) où sont affichées les données de manière lisible et organisée.

Fonctions des boutons:

- La fonction **refresh ()** met à jour le tableau affiché dans l'interface pour refléter les dernières informations de la table sélectionnée.
- La fonction **open_insert_window ()** ouvre une nouvelle fenêtre dédiée à l'insertion d'un nouvel enregistrement. Elle récupère automatiquement les colonnes de la table sélectionnée, crée des champs de saisie pour chaque champ modifiable et permet à l'utilisateur de saisir ses données.

- Une fonction interne **on_insert ()** collecte les valeurs entrées, les envoie à la base de données grâce à la fonction **insert_data_dynamic ()**, puis ferme la fenêtre d'insertion et actualise le tableau principal pour que le nouvel enregistrement apparaisse immédiatement.
- La fonction **delete_selected()** permet de supprimer un enregistrement sélectionné dans le tableau. Elle vérifie d'abord qu'une ligne est bien choisie, récupère la clé primaire correspondante, demande une confirmation à l'utilisateur, puis appelle la fonction **delete_row()** pour effectuer la suppression et actualise le tableau via **refresh()**.
- La fonction **open_update_window()** ouvre une fenêtre permettant de modifier un enregistrement existant. Elle affiche les champs de la ligne sélectionnée dans des zones de saisie pré-remplies, puis la fonction interne **on_update()** met à jour la base de données grâce à **update_row()** et rafraîchit le tableau principal après validation.

Conclusion

En conclusion, ce projet nous a permis de développer une application interactive utilisant **Tkinter** pour la gestion efficace des données. Il nous a offert l'opportunité de combiner l'interface graphique avec la gestion d'une base de données, renforçant ainsi nos compétences en programmation et en organisation des informations. Ce travail nous a aussi appris à anticiper les besoins des utilisateurs et à concevoir une solution pratique et fonctionnelle.

Nous adressons nos sincères remerciements à **Mr.Hain** pour son accompagnement, sa patience et ses précieux conseils tout au long de la réalisation de ce projet. Son soutien nous a grandement aidés à surmonter les difficultés techniques et à mener à bien ce travail.