

Université Hassan II de Casablanca

Ecole Nationale Supérieure des Arts et Métiers

Intelligence Artificielle et Génie Informatique

RAPPORT DE PROJET DE PROGRAMMATION ORIENTÉE OBJET EN PYTHON

A decorative scroll box with a light gray background and a dark gray border. The text is centered within the scroll.

**Système de gestion d'un club
universitaire développé en Python
(Tkinter)**

Elaboré par :

- ❖ TELOUANI Amine
- ❖ SBIA Youness

Présenté à :

- ❖ M. Mustapha HAIN

Année Universitaire 2025/2026

SOMMAIRE

I. Présentation du projet :	1
II. Conception du projet :	1
III. Structure et architecture du code :	3
3.1 Organisation des fichiers	3
3.2 Logique POO :	3
3.3 Interaction interface-base de données :	4
IV. Interface graphique :	4
V. Illustrations des opérations et résultats obtenus :	5

LISTE DES FIGURES

Figure 1 : Diagramme de classes représentant les entités Membre, Activité et Participation.	1
Figure 2 : Modèle conceptuel de données des entités Membre, Activité.	2
Figure 3 : Interface principale.	5
Figure 4 : Fenêtre ajouter membre.	6
Figure 5 : Fenêtre ajout participation du membre en activité	6
Figure 6 : Fenêtre ajout activité	6
Figure 7 : Détails du membre	6
Figure 8 : Liste des activités d'un membre	7
Figure 9 : Résultat d'une recherche	7

I. Présentation du projet :

Ce projet consiste à développer une application de gestion de club en utilisant le langage Python et la bibliothèque Tkinter pour la création de l'interface graphique. L'objectif principal est de faciliter la gestion des membres, des activités et de leurs participations au sein du club, en offrant une solution intuitive, interactive et efficace.

L'application permet à l'utilisateur d'ajouter, modifier, supprimer et rechercher des informations relatives aux membres et aux activités. Elle intègre également une gestion des participations, reliant chaque membre aux activités auxquelles il participe, avec la date correspondante.

La base de données utilisée est SQLite, ce qui garantit une gestion locale, simple et légère des données. Le projet met en œuvre les principes de la programmation orientée objet (POO) à travers la création de classes telles que Membre, Activite et Participation, chacune responsable de ses propres opérations CRUD (Create, Read, Update, Delete).

Ce travail illustre concrètement la manière dont la POO et les interfaces graphiques peuvent être combinées pour créer une application complète, fonctionnelle et bien structurée. Tous les codes développés, ainsi que les ajustements et questions rencontrés au cours de la réalisation, s'inscrivent dans cette démarche d'apprentissage et de mise en pratique des concepts de programmation objet.

II. Conception du projet :

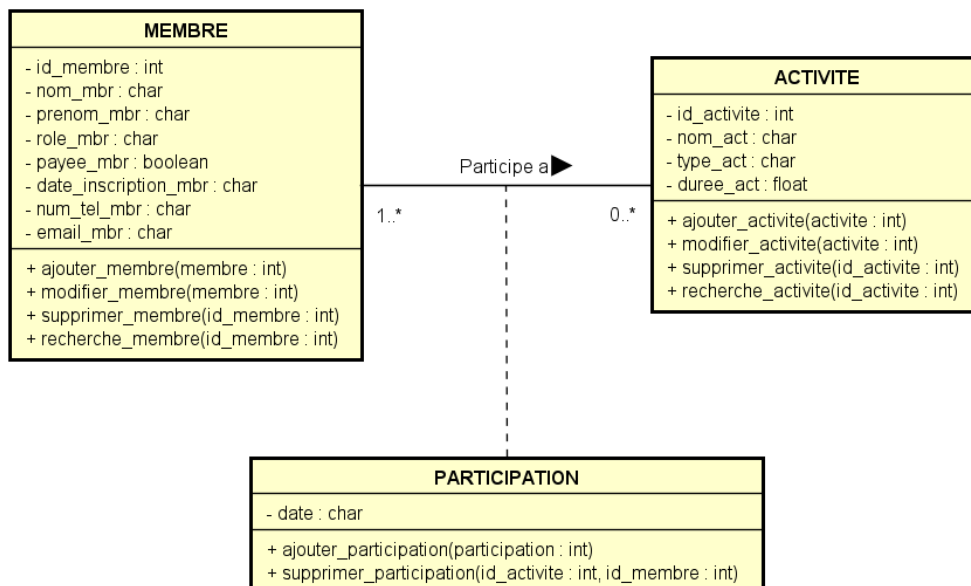


Figure 1 : Diagramme de classes représentant les entités Membre, Activité et Participation.

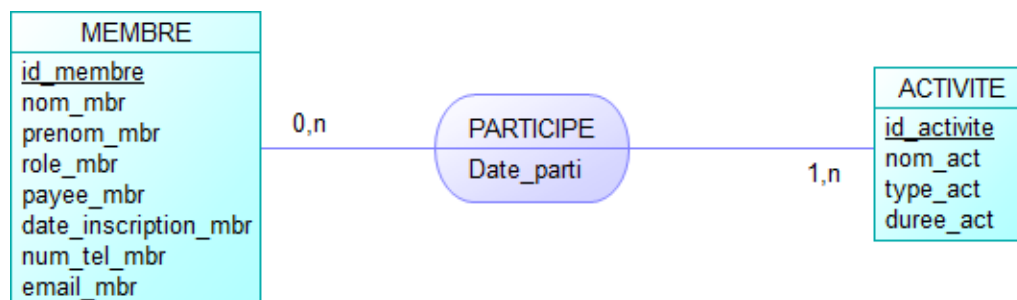


Figure 2 : Modèle conceptuel de données des entités Membre, Activité et leur relation de participation.

Les deux schémas représentent la structure logique et orientée objet du projet de gestion de club, illustrant les relations entre les entités principales : Membre, Activité et Participation.

- Le premier schéma est un diagramme de classes UML qui décrit la structure du code Python. Il montre les attributs et méthodes de chaque classe, ainsi que leurs relations d'association. Chaque classe correspond directement à une table de la base de données SQLite et regroupe les opérations CRUD (ajouter, modifier, supprimer, rechercher). Ce diagramme met en évidence l'approche programmation orientée objet (POO) utilisée dans le projet.
- Le second schéma est un modèle conceptuel de données (MCD). Il illustre les relations entre les tables de la base de données :
 - Un membre peut participer à plusieurs activités,
 - Une activité peut accueillir plusieurs membres.

Cette relation n–n est gérée par la table intermédiaire Participation, qui enregistre aussi la date de participation.

Ensemble, ces deux schémas montrent la cohérence entre la conception objet et la conception relationnelle du projet.

III. Structure et architecture du code :

3.1 Organisation des fichiers

Le projet est structuré en trois fichiers principaux afin de séparer clairement les responsabilités :

a. model :

Ce fichier contient les classes représentant les tables de la base de données SQLite : Membre, Activite et Participation. Chaque classe comporte des méthodes CRUD (Ajouter, Modifier, Supprimer, Rechercher) permettant d'interagir avec la base de données.

b. crud_operations :

Ce fichier contient les fonctions qui créent l'interface graphique pour gérer les membres, les activités et les participations. Il utilise les classes définies dans model.py pour effectuer les opérations CRUD via des fenêtres Tkinter. Chaque fonction correspond à une action utilisateur : ajouter, modifier, supprimer ou rechercher un membre ou une activité, ainsi qu'ajouter une participation liant un membre à une activité.

c. main :

Ce fichier lance l'application et crée la fenêtre principale Tkinter. Il importe les fonctions de crud_operations.py pour afficher les boutons et les champs de recherche, et organise l'interface graphique afin que l'utilisateur puisse interagir avec la base de données. Il sert de point d'entrée et coordonne la communication entre l'interface et les classes de model.py.

3.2 Logique POO :

L'application utilise la programmation orientée objet (POO) pour représenter les entités du club. Chaque classe correspond à une table de la base de données :

- **Membre** : contient les attributs id_membre, nom, prenom, role, payee, date_inscription, tel et les méthodes CRUD (ajouter_membre, modifier_membre, supprimer_membre, recherche_membre).
- **Activite** : contient les attributs id_activite, nom, type, duree et les méthodes CRUD associées.
- **Participation** : relie un membre à une activité via id_membre, id_activite, date_participation et offre des méthodes pour ajouter ou supprimer une participation.

Cette architecture permet de séparer la logique métier de l'interface graphique, de faciliter la maintenance et de réutiliser le code. Les méthodes CRUD encapsulent toutes les interactions avec la base SQLite, assurant un contrôle centralisé des données.

3.3 Interaction interface-base de données :

L'interface graphique Tkinter communique avec la base de données SQLite via les méthodes CRUD des classes Membre, Activite et Participation.

- Les boutons de l'interface déclenchent des fonctions de crud_operations.py qui créent des objets correspondants et appellent les méthodes CRUD.
- Par exemple, cliquer sur Ajouter Membre crée un objet Membre et exécute ajouter_membre() pour insérer les données dans SQLite.
- Les recherches et affichages utilisent les méthodes recherche_membre() ou recherche_activite() pour récupérer les données et les afficher dans la fenêtre.
- La fonction is_table_empty() est utilisée pour activer ou désactiver automatiquement les boutons Modifier/Supprimer selon l'état des tables.

Cette architecture assure un lien direct mais sécurisé entre l'interface et la base, tout en respectant la logique POO.

IV. Interface graphique :

L'application utilise la bibliothèque Tkinter pour fournir une interface graphique simple et intuitive. L'objectif est de permettre à l'utilisateur de gérer facilement les membres, les activités et les participations du club.

a. Fenêtre principale :

La fenêtre principale regroupe tous les boutons et champs de recherche nécessaires à la gestion des données. L'interface est organisée par catégories : Membres, Activités et Participations. Chaque catégorie dispose des boutons pour Ajouter, Modifier, Supprimer et Rechercher.

b. Gestion des Membres :

- **Ajouter un membre :** ouvre une fenêtre avec des champs pour l'ID, le nom, le prénom, le rôle, le statut de paiement, la date d'inscription et le numéro de téléphone.
- **Modifier ou supprimer un membre :** ces boutons sont activés uniquement si la table des membres contient des données, évitant ainsi les erreurs.
- **Recherche :** permet de rechercher un membre par son ID et affiche les informations correspondantes dans une boîte de dialogue.

c. Gestion des Activités :

- **Ajouter, modifier ou supprimer une activité :** des fenêtres similaires à celles des membres permettent de gérer les activités avec les champs ID, nom, type et durée.
- **Recherche :** permet de retrouver une activité par son ID et d'afficher ses détails.

d. Gestion des Participations :

- **Ajouter une participation :** lie un membre à une activité en précisant la date de participation. Cela permet de suivre quelles activités chaque membre a réalisées.

e. Simplicité et ergonomie :

L'interface utilise des boutons clairement étiquetés et des champs de saisie organisés de manière logique. Les états des boutons sont gérés automatiquement pour éviter les opérations invalides lorsque les tables sont vides.

V. Illustrations des opérations et résultats obtenus :

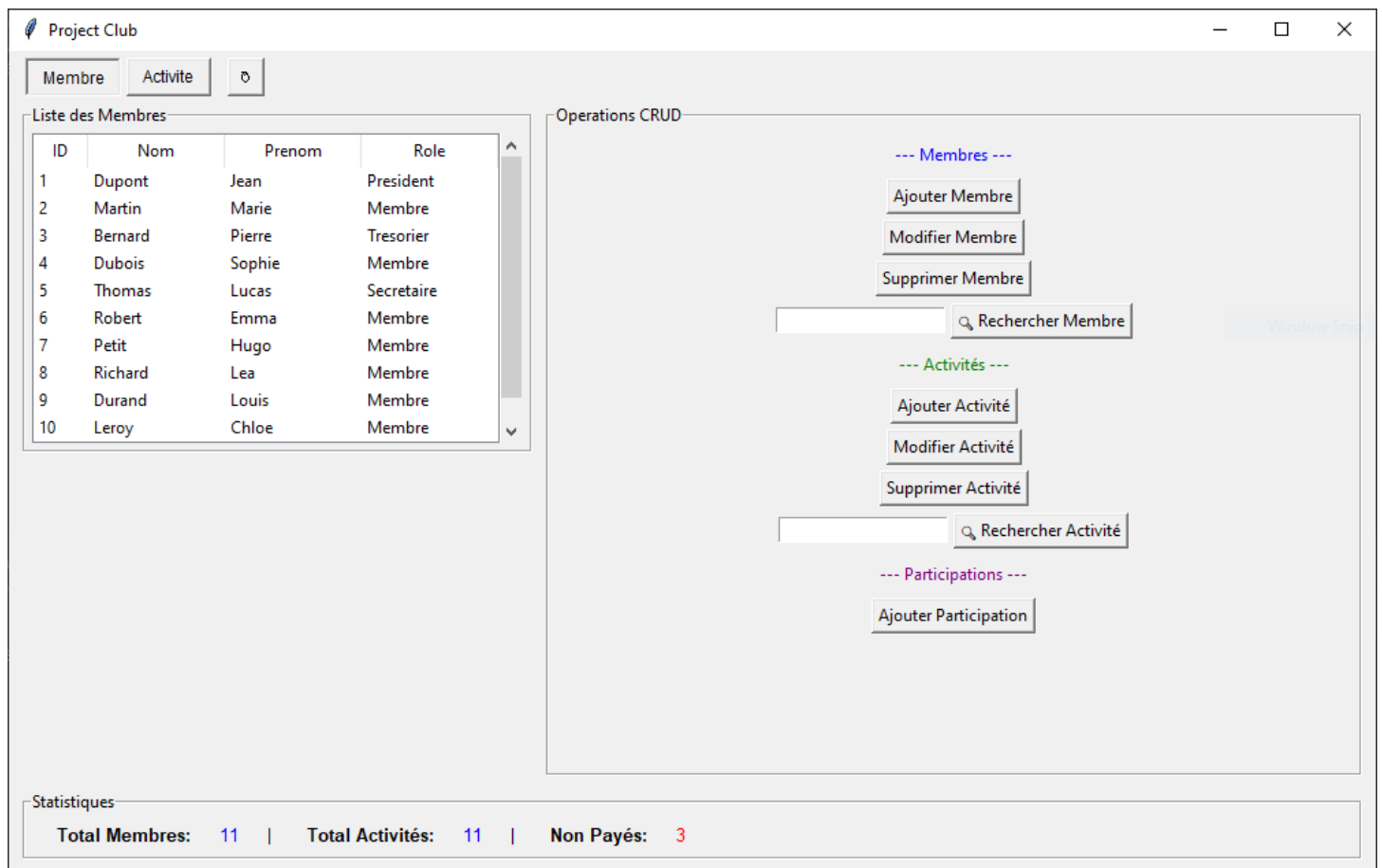
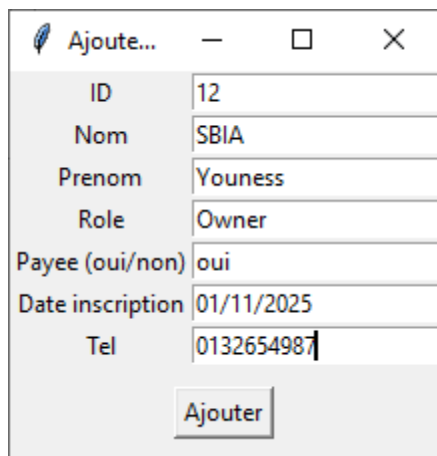


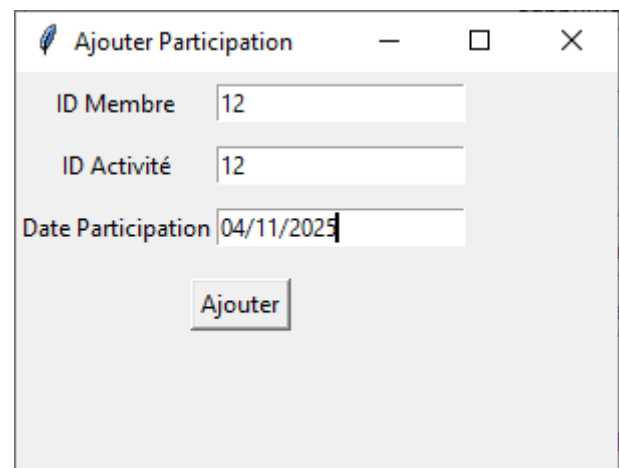
Figure 3 : Interface principale



Ajoute...

ID	12
Nom	SBIA
Prenom	Youness
Role	Owner
Payee (oui/non)	oui
Date inscription	01/11/2025
Tel	0132654987

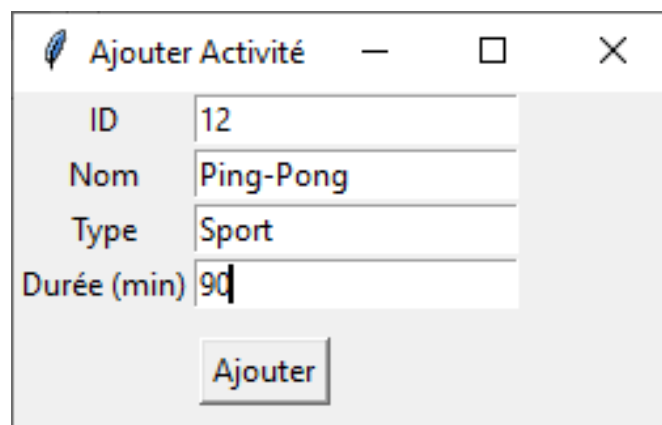
Ajouter

Figure 4 : Fenêtre ajouter membre

Ajouter Participation

ID Membre	12
ID Activité	12
Date Participation	04/11/2025

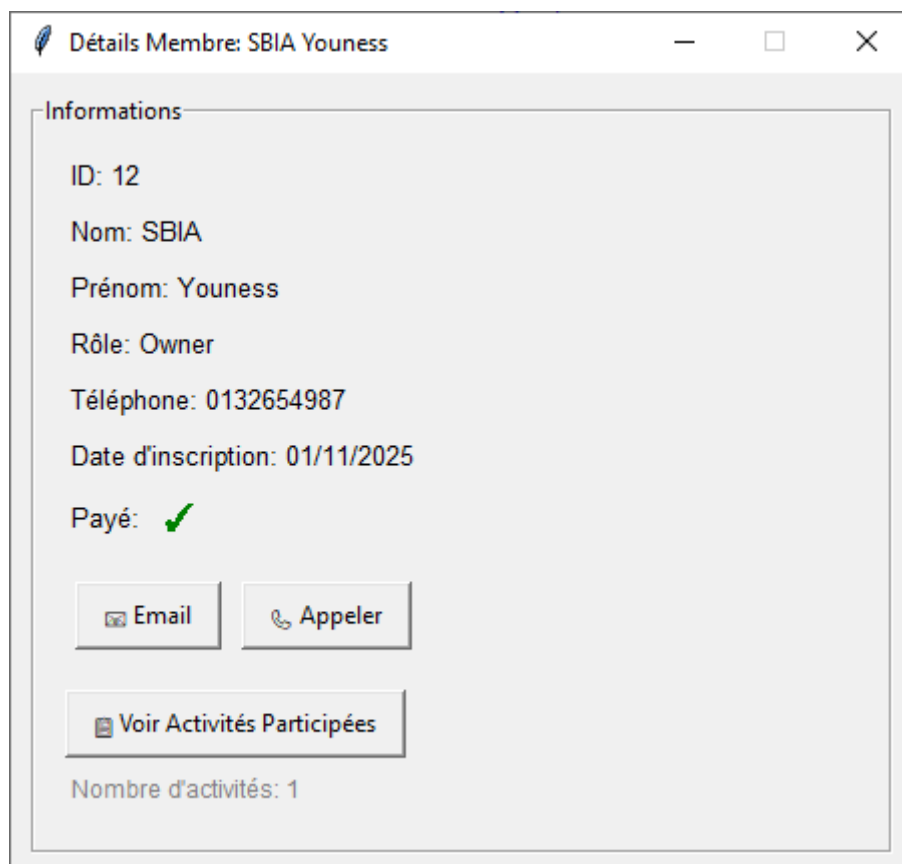
Ajouter

Figure 5 : Fenêtre ajout participation du membre en activité

Ajouter Activité

ID	12
Nom	Ping-Pong
Type	Sport
Durée (min)	90

Ajouter

Figure 6 : Fenêtre ajout activité

Détails Membre: SBIA Youness

Informations

ID: 12
Nom: SBIA
Prénom: Youness
Rôle: Owner
Téléphone: 0132654987
Date d'inscription: 01/11/2025
Payé: ✓

Email Appeler

Voir Activités Participées

Nombre d'activités: 1

Figure 7 : Détails du membre

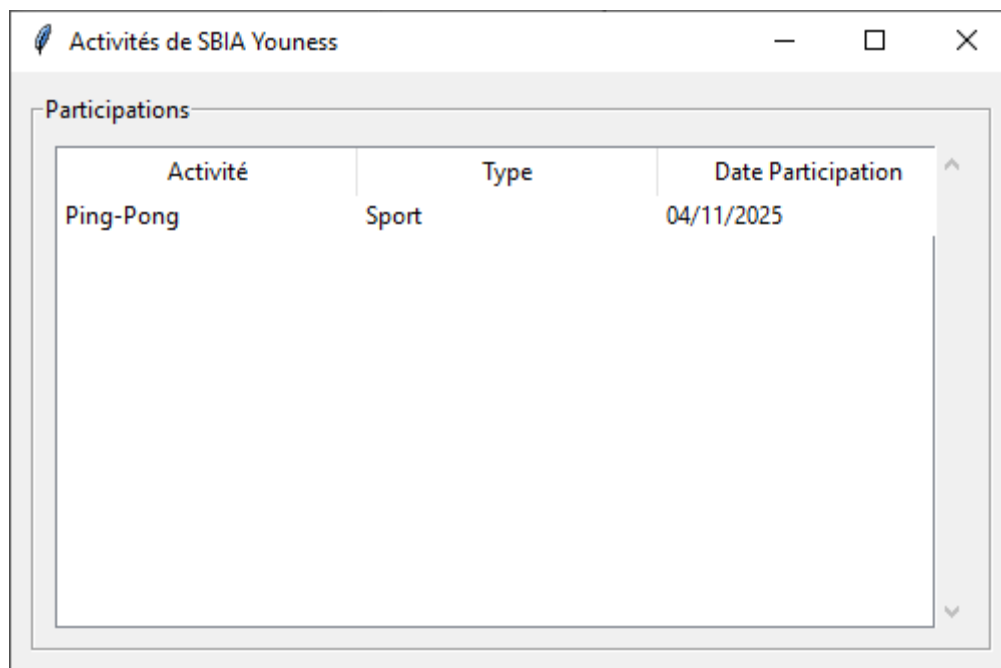


Figure 8 : Liste des activités d'un membre

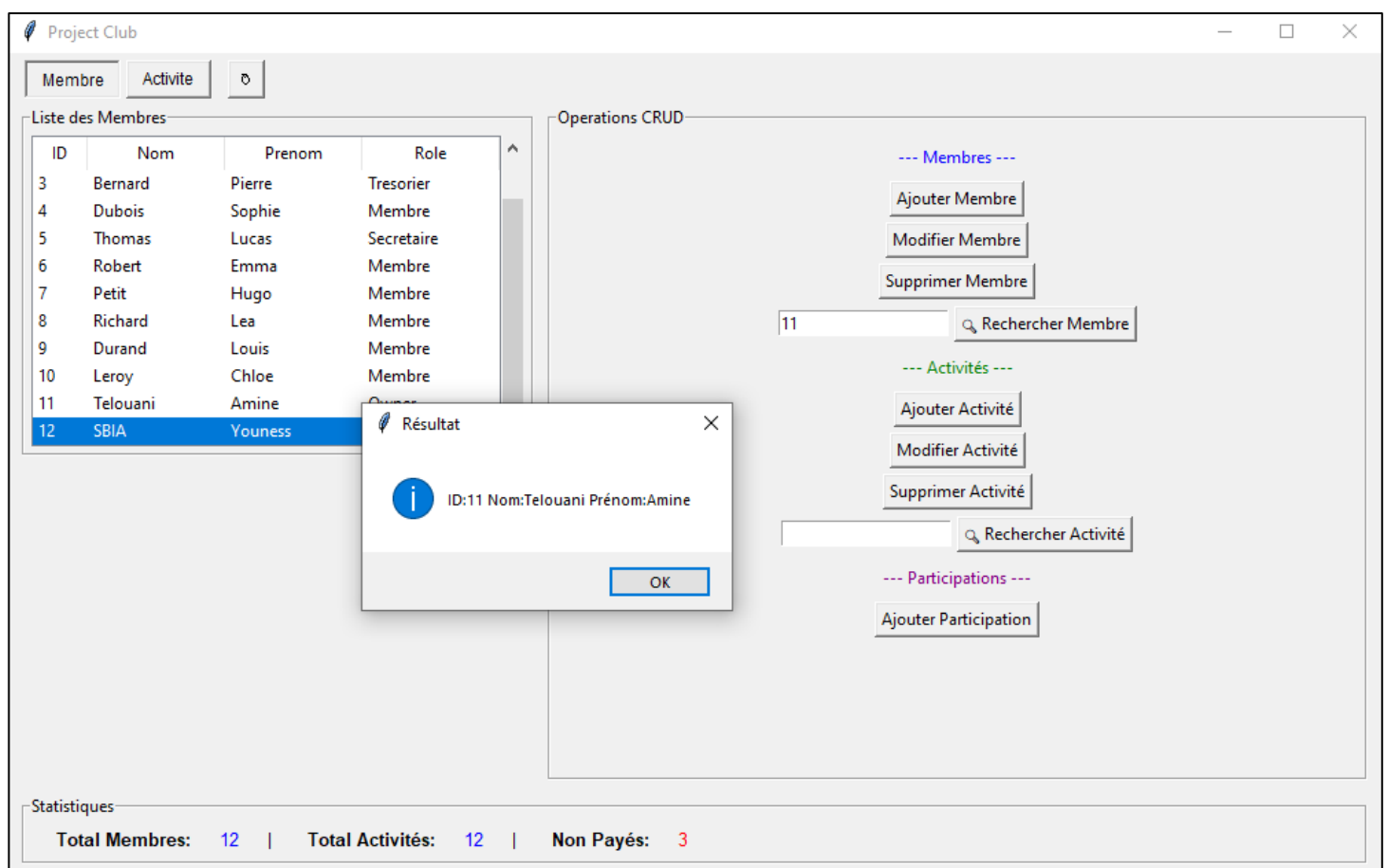


Figure 9 : Résultat d'une recherche