Application Bureautique : STANIA Tkinter

Introduction :

Le présent rapport documente le développement et la mise en œuvre d'une application bureautique de gestion des matchs, conçue dans le cadre du projet STANIA. STANIA est une plateforme web dynamique dédiée à la gestion de matchs, aux paris sportifs et à la gestion de données relatives aux équipes et aux joueurs. Cette suite d'applications vise à faciliter et à améliorer l'expérience des utilisateurs engagés dans l'univers du sport, que ce soit en tant qu'administrateurs et commentateurs.

Le projet STANIA est une initiative globale qui repose sur le développement de multiples solutions logicielles pour répondre aux besoins variés de sa communauté d'utilisateurs. L'application bureautique de gestion des matchs, que nous abordons dans ce rapport, est l'un des composants clés de cette suite. Elle s'adresse principalement aux administrateurs de la plateforme STANIA, leur offrant des outils puissants pour la gestion efficace des matchs sportifs.

Le développement de cette application bureautique est le fruit d'un travail méticuleux qui a mobilisé des compétences techniques en Python et l'utilisation de la bibliothèque Tkinter pour la création de l'interface utilisateur. Dans les sections à suivre, nous explorerons en détail les fonctionnalités de cette application, ses composants techniques, son intégration au sein de l'écosystème STANIA, ainsi que les perspectives d'amélioration pour répondre aux besoins futurs de l'utilisateur.

Ce rapport vise à offrir une vision d'ensemble complète de l'application de gestion des matchs, soulignant son importance dans le contexte de STANIA, et à fournir un guide pour ceux qui souhaitent comprendre son fonctionnement et ses possibilités d'extension.

Tkinter dans les applications bureautiques :

Tkinter, abréviation de "Tk interface", est une bibliothèque Python standard utilisée pour créer des interfaces graphiques utilisateur (GUI) dans les applications bureautiques. Tkinter repose sur le kit de développement d'interface utilisateur Tk, qui a été développé à l'origine pour le langage de programmation Tcl (Tool Command Language). Il fournit un ensemble de modules et de composants permettant aux développeurs de créer des fenêtres, des boutons, des zones de texte, des menus, des boîtes de dialogue et bien d'autres éléments d'interface utilisateur interactifs.

Les applications bureautiques sont des logiciels conçus pour faciliter et améliorer la productivité dans un environnement de travail. Tkinter est largement utilisé dans le développement de telles applications en raison de sa simplicité, de sa polyvalence et de son intégration native avec Python. Voici quelques points clés à retenir concernant l'utilisation de Tkinter dans les applications bureautiques :

1. **Facilité d'utilisation** : Tkinter est apprécié pour sa simplicité, ce qui en fait un choix idéal pour les développeurs qui souhaitent créer des interfaces utilisateur conviviales sans avoir à gérer des détails complexes.

2. **Personnalisation** : Tkinter offre des possibilités de personnalisation importantes, permettant aux développeurs de concevoir des interfaces qui correspondent à l'esthétique de leur application ou de leur entreprise.
3. **Interaction utilisateur** : Les applications bureautiques nécessitent souvent une interaction utilisateur efficace, et Tkinter propose une variété de widgets et d'éléments d'interface pour répondre à ces besoins.
4. **Intégration** : Tkinter s'intègre harmonieusement avec d'autres bibliothèques et frameworks Python, ce qui facilite l'ajout d'interfaces graphiques à des applications existantes.
5. **Portabilité** : Les applications développées avec Tkinter sont portables, ce qui signifie qu'elles peuvent être exécutées sur différentes plates-formes sans modifications significatives.

Dans le contexte de notre application de gestion des matchs pour STANIA, l'utilisation de Tkinter nous a permis de créer une interface utilisateur conviviale, tout en facilitant la gestion des matchs par les administrateurs de la plateforme.

Dans les sections suivantes de ce rapport, nous explorerons plus en détail comment Tkinter a été utilisé pour créer l'interface de l'application bureautique et comment cela a contribué à la réussite de notre projet.

Introduction à l'utilisation de PyCharm pour le développement de l'application bureautique

Le choix d'un environnement de développement intégré (IDE) joue un rôle crucial dans le processus de création d'une application bureautique. PyCharm, développé par JetBrains, est devenu un choix populaire parmi les développeurs Python en raison de ses fonctionnalités avancées, de son support complet de Python, et de sa facilité d'utilisation. Ce rapport examine l'utilisation de PyCharm comme éditeur principal pour le développement de notre application bureautique de gestion des matchs dans le contexte de la suite STANIA.

PyCharm offre un ensemble d'outils puissants qui facilitent la conception, le débogage et la gestion de projets Python, ce qui en fait un choix judicieux pour les développeurs soucieux de la qualité et de la productivité. Dans cette section, nous explorerons comment PyCharm a été utilisé pour optimiser le processus de développement de notre application bureautique.

Nous discuterons des caractéristiques et des avantages spécifiques de PyCharm, notamment :

1. **Intégration de Python** : PyCharm offre une intégration transparente avec Python, facilitant la gestion des dépendances, la création d'environnements virtuels, et l'exécution de scripts Python directement depuis l'IDE.
2. **Assistance à la programmation** : L'IDE propose un ensemble d'outils pour l'autocomplétion, la vérification statique, et la suggestion de code, ce qui accélère le processus de développement et améliore la qualité du code.
3. **Débogage avancé** : PyCharm permet un débogage avancé, offrant des fonctionnalités telles que le débogage à distance, le suivi des variables, et la possibilité de fixer des points d'arrêt pour détecter les erreurs plus rapidement.

4. **Gestion de projet** : L'IDE simplifie la gestion de projets, y compris le suivi des versions, l'intégration avec des systèmes de contrôle de code source, et des outils de gestion de paquets.
5. **Interface conviviale** : PyCharm offre une interface utilisateur intuitive, personnalisable et adaptable aux préférences de chaque développeur, améliorant ainsi le confort et la productivité de l'utilisateur.

Dans les sections suivantes de ce rapport, nous examinerons comment PyCharm a été utilisé pour le développement de notre application bureautique, en mettant en évidence les fonctionnalités.

Exemple d'une fenêtre Tkinter :

Utilisation de Pycharm interface

- Le code de Creation de fenetre Tkinter()

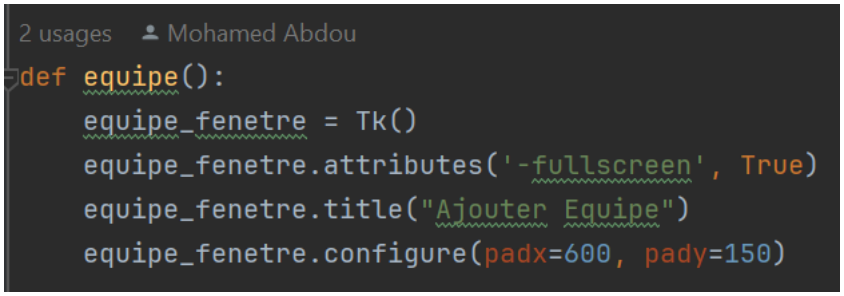


Figure 1: Exemple de creation d'une fenetre Tkinter

But de l'Application Bureautique de Gestion des Matchs :

L'application bureautique que nous avons développée est un outil destiné à simplifier la gestion des données des matchs sportifs dans la base de données de la suite STANIA. Son rôle principal est de permettre aux administrateurs d'effectuer des opérations CRUD (Créer, Lire, Mettre à jour, Supprimer) sur les enregistrements de matchs. Plus spécifiquement, ses fonctions clés incluent :

1. **Création de Matchs** : Les administrateurs peuvent ajouter de nouveaux matchs à la base de données en spécifiant des détails tels que la date, l'heure, les équipes participantes et le lieu.
2. **Consultation des Matchs** : L'application offre aux administrateurs un moyen convivial de consulter les informations existantes sur les matchs enregistrés, fournissant un aperçu rapide et complet des données de match.
3. **Mise à Jour des Matchs** : Les administrateurs ont la possibilité de mettre à jour les enregistrements de matchs existants, que ce soit pour corriger des erreurs, mettre à jour les scores, ou modifier d'autres détails pertinents.
4. **Suppression de Matchs** : En cas de besoin, les administrateurs peuvent supprimer des matchs de la base de données, garantissant ainsi la précision et la pertinence des données.

L'objectif principal de cette application est de simplifier et d'automatiser le processus de gestion des matchs pour les administrateurs, en éliminant la nécessité d'intervenir directement dans la base de données ou d'utiliser des requêtes SQL complexes. Elle vise à améliorer l'efficacité de la gestion tout en minimisant les risques d'erreurs humaines.

Dans les sections suivantes de ce rapport, nous examinerons en détail comment l'application a été conçue pour atteindre ces objectifs, y compris les fonctionnalités spécifiques qui facilitent les opérations CRUD sur les matchs de données.

Réalisation et mise en œuvre :

La partie code sur Tkinter :

La base de données et la création de la table des matchs avec **SQLite3** :

SQLite3 est un système de gestion de base de données relationnelle (SGBDR) intégré à Python. Il offre une solution légère et simple pour la gestion des données, stockées dans un fichier unique, ce qui en fait un choix idéal pour les applications locales ou embarquées. Il prend en charge la création, la modification, la lecture et la suppression de données de manière efficace, offrant une compatibilité SQL complète.

Code sur Pycharm :

1- Base de donnees :
Le fichier sql.py de mon projet, contient les differents requetes pour la creation de mes differents tables dans la base de donnees qui est dans mon projet sous format d'un fichier STANIA.db, ce fichier pendant, la realisation de mon projet, est inserer dans le logiciel « **DB Browser** » (figure ci-dessous).

Figure 2: DB Browser (MySqLite)

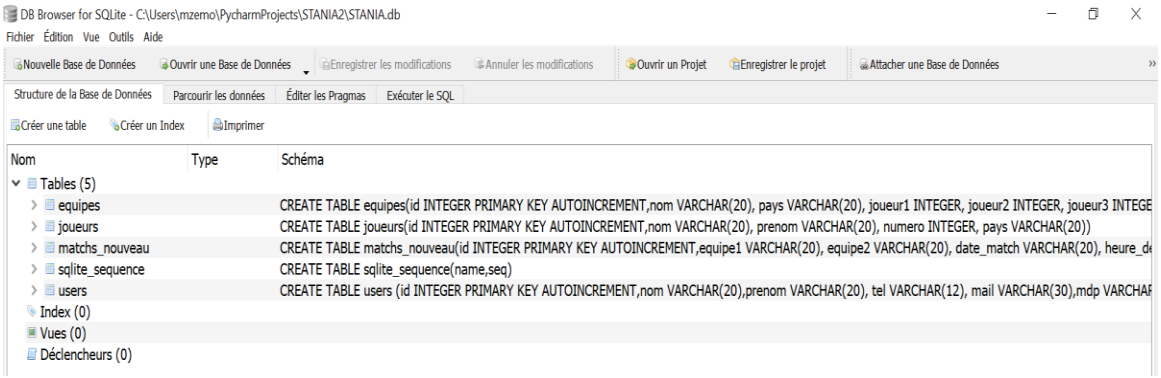


Figure 3: La base de donnees et table Sql de STANIA

Contenu du fichier sql.py

```
import sqlite3

connection = sqlite3.connect("STANIA.db")
cursor = connection.cursor()
```

Figure 4 : Cursor, pour la creation des requetes sql

```
cursor.execute("CREATE TABLE IF NOT EXISTS "
               "joueurs(id INTEGER PRIMARY KEY AUTOINCREMENT,"
               "nom VARCHAR(20), "
               "prenom VARCHAR(20), "
               "numero INTEGER, "
               "pays VARCHAR(20))")
```

Figure 5 : Creation de la table Joueur

```
cursor.execute("CREATE TABLE IF NOT EXISTS "
               "equipes(id INTEGER PRIMARY KEY AUTOINCREMENT,"
               "nom VARCHAR(20), "
               "pays VARCHAR(20), "
               "joueur1 INTEGER, joueur2 INTEGER, joueur3 INTEGER, "
               "joueur4 INTEGER, joueur5 INTEGER, joueur6 INTEGER, "
               "joueur7 INTEGER)")
```

Figure 6: Creation de la table Equipes

```
cursor.execute("CREATE TABLE IF NOT EXISTS "
               "matchs_nouveau(id INTEGER PRIMARY KEY AUTOINCREMENT, "
               "equipe1 VARCHAR(20), "
               "equipe2 VARCHAR(20), "
               "date_match VARCHAR(20), "
               "heure_debut VARCHAR(20), "
               "heure_fin VARCHAR(20), "
               "cote_match VARCHAR(20), "
               "nombre_paris VARCHAR(20), "
               "lieu VARCHAR(20), "
               "etat_match VARCHAR(20))")
```

Figure 7: Creation de la table Matchs

```
cursor.execute("CREATE TABLE IF NOT EXISTS "
               "users (id INTEGER PRIMARY KEY AUTOINCREMENT,"
               "nom VARCHAR(20),"
               "prenom VARCHAR(20), "
               "tel VARCHAR(12), "
               "mail VARCHAR(30),"
               "mdp VARCHAR(20))")
```

Figure 8: Creation de la table Users

Le fichier ***sql.py*** contient les différentes opérations sur la table des Joueurs :

- L'insertion
- La sélection
- La suppression

```
2 usages  Mohamed Abdou *
def inserer_joueur(*args):
    cursor.execute( _sql: " INSERT INTO joueurs(nom, prenom, numero, pays) "
                   "VALUES ( ?, ?, ?, ?) ", args)
    connection.commit()

2 usages  Mohamed Abdou *
def supprimer_joueur(*args):
    cursor.execute( _sql: "DELETE FROM joueurs WHERE id=?", args)
    connection.commit()

2 usages  new *
def liste_joueur():
    cursor.execute("SELECT * FROM joueurs")
    return cursor.fetchall()
```

Figure 9 : Fonctions sur la table Joueurs

Le formulaire de l'ajout d'un joueur :

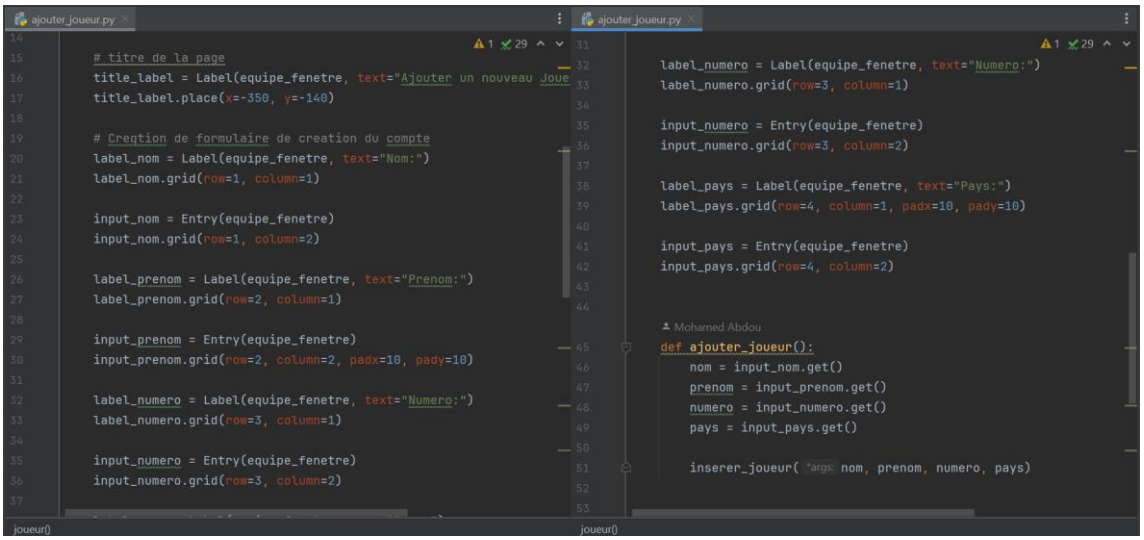


Figure 10: Formulaire Tkinter pour ajouter un Joueur

Création de la table des Equipes :

Établissement de la connexion avec la base de données
MySqlite3

Le fichier ***Sql.py*** contient les différentes opérations sur la table des Equipes:

- L'insertion
- La sélection
- La suppression

```
2 usages   Mohamed Abdou *
def inserer_equipe(*args):
    cursor.execute( _sql: " INSERT INTO equipes(nom, pays, joueur1, joueur2, "
                    "joueur3, joueur4, joueur5, joueur6, joueur7) "
                    "VALUES ( ?, ?, ?, ?, ?, ?, ?, ?, ?) ", args)
    connection.commit()

2 usages   new *
def supprimer_equipe(*args):
    cursor.execute( _sql: "DELETE FROM equipes WHERE id=?", args)
    connection.commit()

2 usages   new *
def liste_equipe():
    cursor.execute("SELECT * FROM equipes")
    return cursor.fetchall()
```

Figure 11: Fonctions sur la table Equipe

Le formulaire de l'ajout d'une équipe :

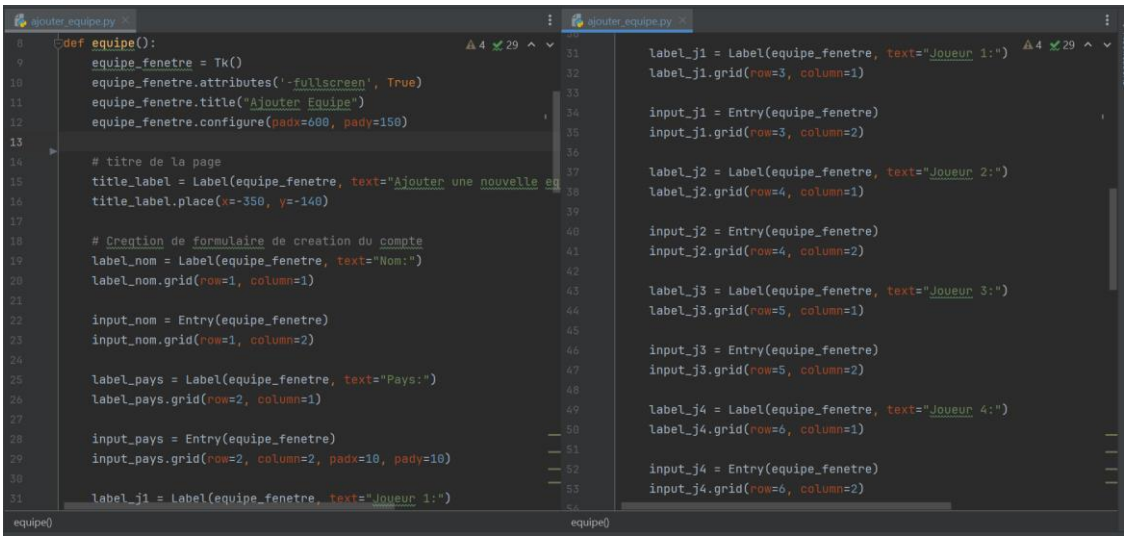


Figure 12: Formulaire Tkinter pour ajouter un Joueur

Création de la table des matchs :

Établissement de la connexion avec la base de données
MySqlite3

Le fichier ***Sql.py*** contient les différentes opérations sur la table des matchs :

- L'insertion
- La sélection
- La suppression

```
2 usages   Mohamed Abdou *
def inserer_match(*args):
    cursor.execute( _sql: " INSERT INTO matchs_nouveau(equipe1, equipe2, "
                    "date_match, heure_debut, heure_fin, cote_match, "
                    "nombre_paris,lieu, etat_match) "
                    "VALUES ( ?, ?, ?, ?, ?, ?, ?, ?, ?) ", args)
    connection.commit()

2 usages   Mohamed Abdou
def supprimer_match(*args):
    cursor.execute( _sql: "DELETE FROM matchs_nouveau WHERE id=?", args)
    connection.commit()

2 usages   Mohamed Abdou
def liste_match():
    cursor.execute("SELECT * FROM matchs_nouveau")
    return cursor.fetchall()
```

Figure 13 : Fonctions sur la table des MAtchs

Le formulaire de l'ajout d'un match :

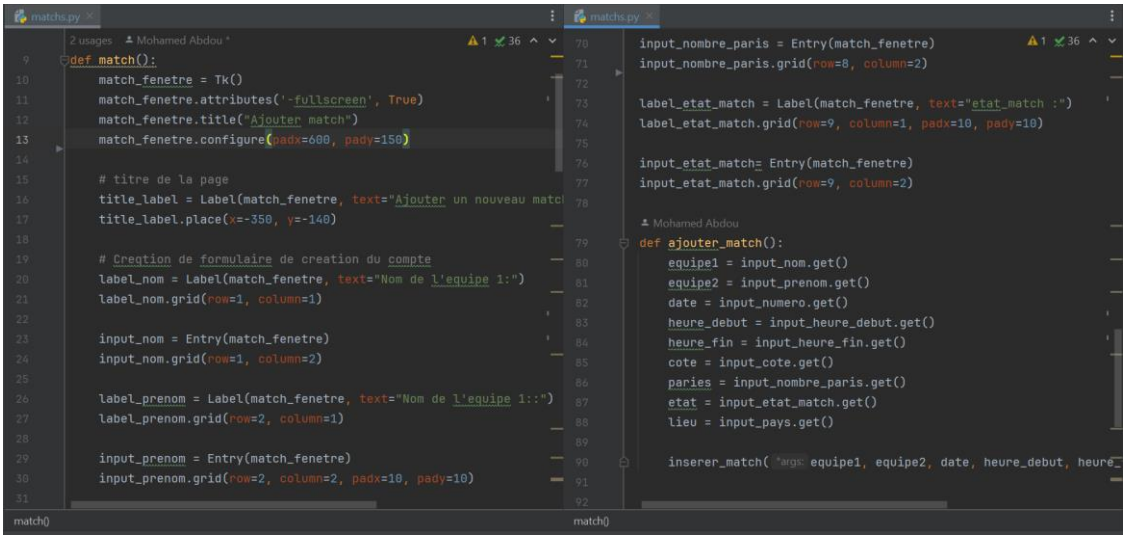


Figure 14: Formulaire Tkinter pour ajouter un Match

Le formulaire de login.py

Figure 15: Formulaire du Login Page

Captures de pages de l'application :

Page d'accueil - Création de compte :

Cette page permet aux administrateurs de créer un compte utilisateur afin d'accéder à l'application. Une fois le compte créé, les administrateurs peuvent ajouter des joueurs, des équipes, et des matchs dans la base de données.

Après la création du compte, l'utilisateur doit cliquer sur le bouton **"Se connecter"** pour accéder à la page de connexion et entrer ses identifiants afin de se connecter à l'application.

Page de connexion :

Sur cette page, l'utilisateur est invité à saisir son adresse e-mail et le mot de passe qu'il a précédemment créés. En cliquant sur le bouton **"Se connecter"**, l'utilisateur peut accéder aux différentes fonctionnalités de l'application bureautique.

Tableau de bord de l'application :

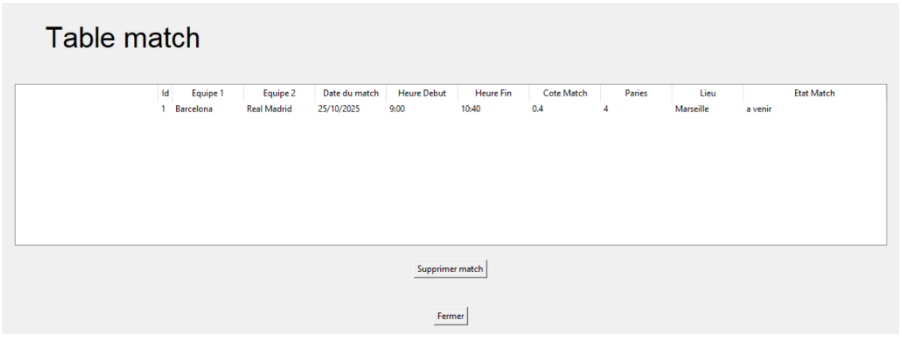
Le tableau de bord offre aux commentateurs un accès centralisé aux différentes entités de l'application, telles que les **matchs**, les **joueurs** et les **équipes**. Depuis cet espace, ils peuvent visualiser et consulter les informations détaillées, notamment celles liées aux matchs en cours ou à venir.

Page d'ajout d'un joueur :

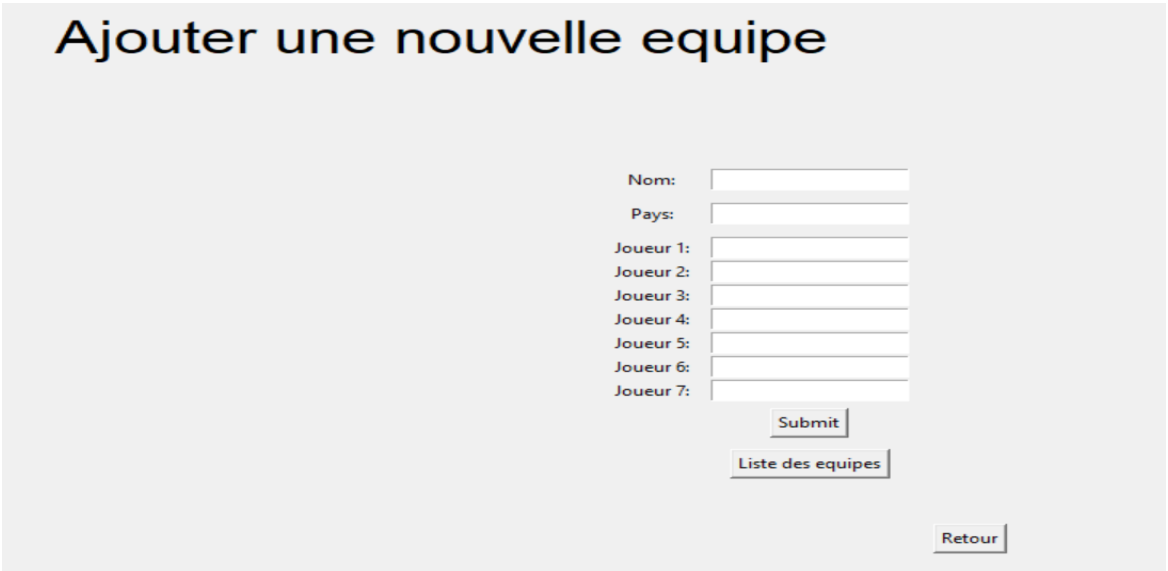
Cette page permet à l'administrateur d'ajouter un joueur à l'application. En cliquant sur **"Liste des joueurs"**, l'administrateur est redirigé vers un tableau (TreeView) qui affiche les différentes informations concernant chaque joueur.

Page d'ajout d'un match :

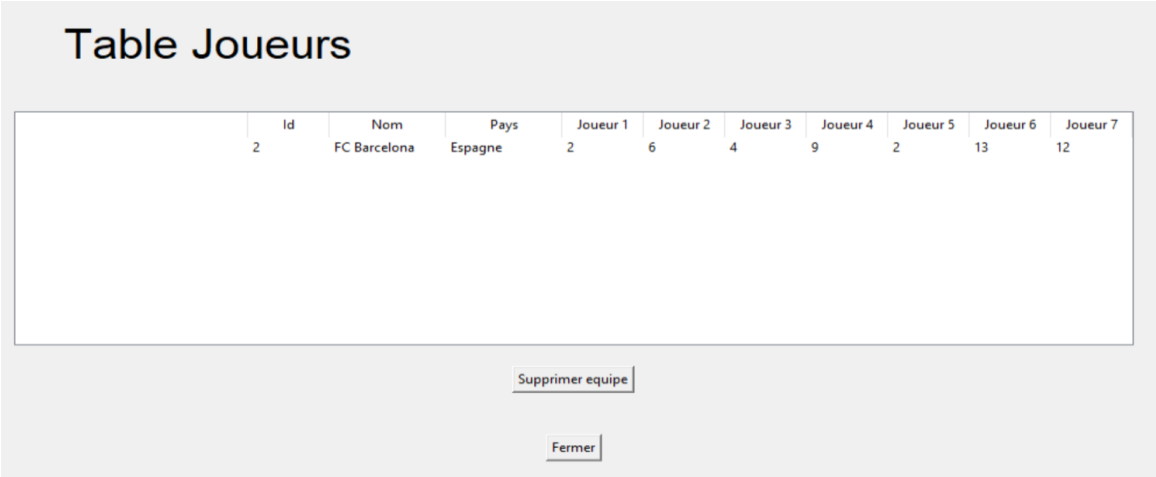
Cette page permet à l'administrateur d'ajouter un match dans l'application. En cliquant sur **"Liste des matchs"**, l'administrateur est redirigé vers un tableau (TreeView) qui affiche les différentes informations concernant chaque match.



Page d'ajout d'une équipe :



Sur cette page, l'administrateur peut ajouter une équipe à l'application. En cliquant sur **"Liste des équipes"**, il est redirigé vers un tableau (TreeView) qui affiche les informations détaillées sur chaque équipe.



En cliquant sur supprimer :

Pour supprimer une ligne dans la table de l'application, il suffit de sélectionner la ligne souhaitée et de cliquer sur le bouton **Supprimer**.

Conclusion

Lorsqu'il s'agit de développer une application bureautique avec Tkinter, cette bibliothèque offre une solution robuste pour créer des interfaces utilisateur conviviales. Tkinter, étant la bibliothèque GUI standard de Python, présente plusieurs avantages significatifs pour le développement d'applications bureautiques.

Tout d'abord, Tkinter est facile à prendre en main, ce qui en fait un excellent choix pour les débutants en programmation. Sa simplicité permet de créer rapidement des interfaces utilisateur fonctionnelles sans une courbe d'apprentissage abrupte. En utilisant Tkinter, il est possible de concevoir des applications bureautiques pour la gestion de documents, de données, des tableurs, des éditeurs de texte, et bien plus encore.

De plus, Tkinter est livré avec une vaste bibliothèque de widgets prédéfinis, tels que des boutons, des listes, des champs de texte, des menus, etc., facilitant ainsi la création d'interfaces riches et interactives. Ces éléments peuvent être agencés et personnalisés pour répondre aux besoins spécifiques de l'application bureautique en cours de développement.

Tkinter facilite également l'intégration avec d'autres bibliothèques et modules Python, ce qui permet d'ajouter des fonctionnalités avancées à l'application bureautique. De plus, étant une bibliothèque standard, Tkinter est généralement déjà installé avec Python, éliminant ainsi le besoin de téléchargement ou d'installation supplémentaire.

En somme, Tkinter apporte une solution simple et efficace pour le développement d'applications bureautiques avec Python. Son ensemble d'outils conviviaux et sa polyvalence en font un choix judicieux pour créer des logiciels de bureau destinés à des tâches variées, offrant une base solide pour les développeurs cherchant à construire des applications bureautiques fonctionnelles et intuitives.