

Application Bureautique : STANIA Tkinter

Lien : https://github.com/abdoma-git/Devoir_APP_Bureautique.git

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()



```
2 usages  Mohamed Abdou
def equipe():
    equipe_fenetre = Tk()
    equipe_fenetre.attributes('-fullscreen', True)
    equipe_fenetre.title("Ajouter Equipe")
    equipe_fenetre.configure(padx=600, pady=150)
```

Figure 1: Exemple de creation d'une fenetre Tkinter

But de l'Application Bureautique de Gestion des Matches :

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 Matches** : 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 Matches** : 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 Matches** : 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 Matches** : 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 données :

Le fichier sql.py de mon projet, contient les différents requetes pour la creation de mes différents **tables** dans la base de données 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).

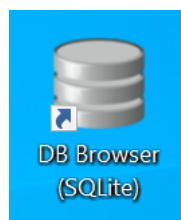


Figure 2: DB Browser (MySqlite)

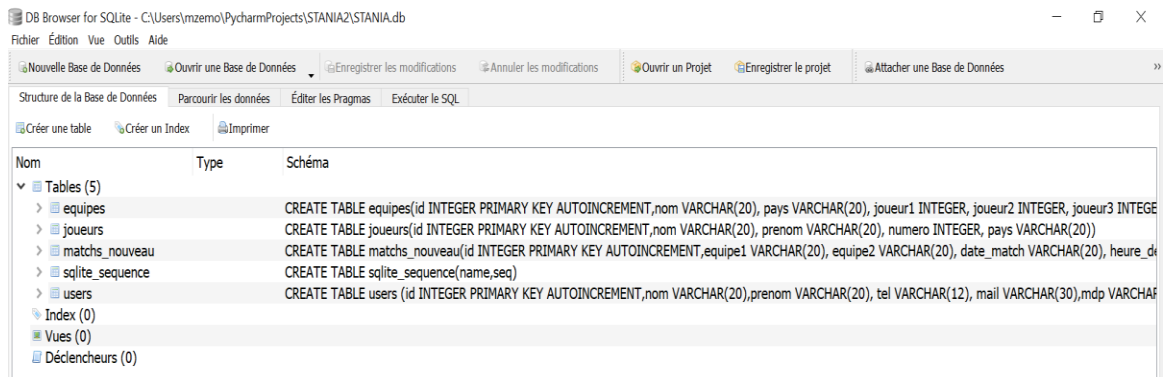


Figure 3: La base de donnees et table Sql de STANIA

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 "
               "matches_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 Matches

```

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 :

```

14 # titre de la page
15 title_label = Label(equipe_fenetre, text="Ajouter un nouveau Joueur")
16 title_label.place(x=350, y=140)
17
18 # Creation de formulaire de creation du compte
19 label_nom = Label(equipe_fenetre, text="Nom:")
20 label_nom.grid(row=1, column=1)
21
22 input_nom = Entry(equipe_fenetre)
23 input_nom.grid(row=1, column=2)
24
25 label_prenom = Label(equipe_fenetre, text="Prenom:")
26 label_prenom.grid(row=2, column=1)
27
28 input_prenom = Entry(equipe_fenetre)
29 input_prenom.grid(row=2, column=2, padx=10, pady=10)
30
31 label_numero = Label(equipe_fenetre, text="Numero:")
32 label_numero.grid(row=3, column=1)
33
34 input_numero = Entry(equipe_fenetre)
35 input_numero.grid(row=3, column=2)
36
37 label_pays = Label(equipe_fenetre, text="Pays:")
38 label_pays.grid(row=4, column=1, padx=10, pady=10)
39
40 input_pays = Entry(equipe_fenetre)
41 input_pays.grid(row=4, column=2)
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

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 insérer_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 :

```
ajouter_equipe.py x
8 def equipe():
9     equipe_fenetre = Tk()
10    equipe_fenetre.attributes('-fullscreen', True)
11    equipe_fenetre.title("Ajouter Equipe")
12    equipe_fenetre.configure(padx=600, pady=150)
13
14    # titre de la page
15    title_label = Label(equipe_fenetre, text="Ajouter une nouvelle équipe")
16    title_label.place(x=-350, y=-140)
17
18    # Creation de formulaire de creation du compte
19    label_nom = Label(equipe_fenetre, text="Nom:")
20    label_nom.grid(row=1, column=1)
21
22    input_nom = Entry(equipe_fenetre)
23    input_nom.grid(row=1, column=2)
24
25    label_pays = Label(equipe_fenetre, text="Pays:")
26    label_pays.grid(row=2, column=1)
27
28    input_pays = Entry(equipe_fenetre)
29    input_pays.grid(row=2, column=2, padx=10, pady=10)
30
31    label_j1 = Label(equipe_fenetre, text="Joueur 1:")
32    label_j1.grid(row=3, column=1)
33
34    input_j1 = Entry(equipe_fenetre)
35    input_j1.grid(row=3, column=2)
36
37    label_j2 = Label(equipe_fenetre, text="Joueur 2:")
38    label_j2.grid(row=4, column=1)
39
40    input_j2 = Entry(equipe_fenetre)
41    input_j2.grid(row=4, column=2)
42
43    label_j3 = Label(equipe_fenetre, text="Joueur 3:")
44    label_j3.grid(row=5, column=1)
45
46    input_j3 = Entry(equipe_fenetre)
47    input_j3.grid(row=5, column=2)
48
49    label_j4 = Label(equipe_fenetre, text="Joueur 4:")
50    label_j4.grid(row=6, column=1)
51
52    input_j4 = Entry(equipe_fenetre)
53    input_j4.grid(row=6, column=2)
54
55    equipe_fenetre.mainloop()
```

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 :

```
match0.py
def match():
    match_fenetre = Tk()
    match_fenetre.attributes('-fullscreen', True)
    match_fenetre.title("Ajouter match")
    match_fenetre.configure(padx=600, pady=150)

    # titre de la page
    title_label = Label(match_fenetre, text="Ajouter un nouveau match")
    title_label.place(x=-350, y=-140)

    # Creation de formulaire de creation du compte
    label_nom = Label(match_fenetre, text="Nom de l'equipe 1:")
    label_nom.grid(row=1, column=1)

    input_nom = Entry(match_fenetre)
    input_nom.grid(row=1, column=2)

    label_prenom = Label(match_fenetre, text="Nom de l'equipe 2:")
    label_prenom.grid(row=2, column=1)

    input_prenom = Entry(match_fenetre)
    input_prenom.grid(row=2, column=2, padx=10, pady=10)

    input_nombre_paris = Entry(match_fenetre)
    input_nombre_paris.grid(row=8, column=2)

    label_etat_match = Label(match_fenetre, text="etat_match :")
    label_etat_match.grid(row=9, column=1, padx=10, pady=10)

    input_etat_match = Entry(match_fenetre)
    input_etat_match.grid(row=9, column=2)

    Mohamed Abdou
    def ajouter_match():
        equipe1 = input_nom.get()
        equipe2 = input_prenom.get()
        date = input_numero.get()
        heure_debut = input_heure_debut.get()
        heure_fin = input_heure_fin.get()
        cote = input_cote.get()
        paries = input_nombre_paris.get()
        etat = input_etat_match.get()
        lieu = input_pays.get()

        inserer_match(*args equipe1, equipe2, date, heure_debut, heure_
```

Figure 14: Formulaire Tkinter pour ajouter un Match

Le formulaire de login.py



```
login_form.py
23 label_mail = Label(login_fenetre, text="Email:")
24 label_mail.grid(row=1, column=1)
25
26 input_mail = Entry(login_fenetre)
27 input_mail.grid(row=1, column=2)
28
29 label_mdp = Label(login_fenetre, text="Mot de passe:")
30 label_mdp.grid(row=2, column=1)
31
32 input_mdp = Entry(login_fenetre, show="*")
33 input_mdp.grid(row=2, column=2, padx=10, pady=10)
34
35 Mohamed Abdou
36 def login():
37     liste = liste_users()
38     mail = input_mail.get()
39     mdp = input_mdp.get()
40
41     for user in liste:
42         if (user[4] == mail and user[5] == mdp):
43             return open_dashboard(login_fenetre)
44         else:
45             label_info = Label(login_fenetre, text="Utilisateur n'existe pas !!")
46             label_info.grid(row=5, column=2)
47
48 se_connecter()
```

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 **matches**, 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 :

The form is titled 'Ajouter un nouveau Joueur'. It contains four input fields labeled 'Nom:', 'Prenom:', 'Numero:', and 'Pays:'. Below these fields are three buttons: 'Submit', 'Liste des joueurs', and 'Retour'.

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.

	Id	Nom	Prenom	Numero	Pays
1	Cristiano	Ronaldo	7	Portugal	

Buttons: Supprimer joueur, Fermer

Page d'ajout d'un match :

Ajouter un nouveau match

Nom de l'equipe 1:

Nom de l'equipe 1:

Date du Match:

Lieu du Match:

Heure de debut :

Heure de fin :

cote_match :

nombre-paris :

etat_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.

Table match

Id	Equipe 1	Equipe 2	Date du match	Heure Debut	Heure Fin	Cote Match	Paries	Lieu	Etat Match
1	Barcelona	Real Madrid	25/10/2025	9:00	10:40	0.4	4	Marseille	a venir

Page d'ajout d'une équipe :

Ajouter une nouvelle equipe

Nom:

Pays:

Joueur 1:

Joueur 2:

Joueur 3:

Joueur 4:

Joueur 5:

Joueur 6:

Joueur 7:

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.

	Id	Nom	Pays	Joueur 1	Joueur 2	Joueur 3	Joueur 4	Joueur 5	Joueur 6	Joueur 7
	2	FC Barcelona	Espagne	2	6	4	9	2	13	12

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.