Sistema de Registro de Dados da Copa do Mundo de Futebol 2022

Luis Felipe Pereira de Carvalho

Departamento de Ciências Exatas - Engenharia de Computação

Universidade Estadual de Feira de Santana (UEFS)
Feira de Santana – BA – Brazil

luisoftbr@outlook.com.br

Abstract. *Looking for ways to follow the games and results of the 2022 football world cup, Zé do Gol, ex-player of the soccer team Fluminense de Feira Futebol Clube proposed to the students of the Computer Engineering Course from the Universidade Estadual de Feira de Santana the creation of a system to register teams, games, results and statistics. The proposed system provokes the creation of a source code for the development of its interface and register in files. For the development, the python language was used. In conclusion, the system does not meet all the requirements that were requested.*

Resumo. *Buscando maneiras para o acompanhamento dos jogos e resultados dos jogos da copa do mundo de futebol de 2022, Zé do Gol, ex-jogador do time de futebol Fluminense de Feira Futebol Clube propôs aos Alunos do Curso de Engenharia de Computação da Universidade Estadual de Feira de Santana a criação de um sistema de cadastramento de seleções, jogos, resultados e estatísticas. O sistema proposto provoca a criação de um código fonte para o desenvolvimento de sua interface e registro em arquivos. Para o desenvolvimento, foi utilizada a linguagem python. Em conclusão, o sistema não cumpre com todos os requisitos que foram solicitados.*

1. Introdução

Segundo Miranda (2022), O futebol é a grande paixão popular do brasileiro. desde criança possuímos o futebol como o principal esporte desenvolvido e vivenciamos esse esporte nas suas mais variadas formas, seja na rua com brincadeira de crianças ou em grande campeonatos como o Brasileirão e a própria copa do mundo.

De acordo com estudos da Nielsen (2018), o Brasil ocupa a 13º posição de países com uma população muito interessada por futebol. Desse modo, é notável que grande parte das pessoas se interessam bastante para o futebol, uma das quais, O Zé do gol, ex-jogador do Fluminense de Feira Futebol Clube, fez uma proposta aos alunos do curso de Engenharia de Computação da UEFS para a criação de um sistema de cadastramento de jogos e seleções da copa do mundo de futebol de 2022 para que ele pudesse estar sempre antenado com os resultados e estatísticas.

Para atender aos requisitos do sistema de cadastramento, o código foi elaborado principalmente com modularização, listas e manipulação de arquivos. A construção do sistema é apresentada na seguinte seção que trata da metodologia utilizada.

2. Metodologia

De maneira a atender e discutir o desenvolvimento do software, foram realizadas sessões Problem-Based-Learning (PBL) para a discussão do sistema e de seus requisitos que foram solicitados e apresentados pelo tutor. Foram realizadas sessões para discutir fatos, ideias, questões e metas para a resolução do sistema.

Para informações técnicas, é necessário informar que o sistema foi desenvolvido integralmente em linguagem Python na versão 3.10. A IDE utilizada foi a Visual Studio Code na versão 1.73.1 no sistema operacional Linux Mint 20.3 Una.

A seguir são apresentados e detalhados os requisitos solicitados para a conclusão do sistema.

2.1 Definição de requisitos

Como citado anteriormente, para a construção do sistema se fez necessário atender alguns requisitos, os quais estão exibidos na tabela 1:

Tabela 1. Requisitos de entrada do cadastro

Requisito	Detalhamento
Cadastro de equipes	Realizar o cadastro das equipes em seus grupos.
Cadastro dos confrontos	Realizar o cadastro dos jogos, informando dia, horário, local, resultado.

A tabela 2 corresponde aos requisitos de processamento dos dados do cadastro.

Tabela 2. Requisitos do cadastro relacionados ao processamento dos dados

Requisito	Detalhamento
Edição de Cadastro	Possibilidade de edição ou exclusão de algum cadastro realizado (equipes, grupos, confrontos)
Resultado dos Confrontos	Realizar o cadastro dos resultados e dos dados referentes aos critérios de desempate dos confrontos realizados.

A tabela 3 corresponde aos requisitos de saída dos dados cadastrados no sistema.

Tabela 3. Requisitos do cadastro relacionados a saída dos dados

Requisito	Detalhamento
Exibição das seleções e grupos	Exibir ao usuário as seleções cadastradas e seus respectivos grupos.
Exibição dos confrontos	Exibir os confrontos de cada grupo.

Exibição dos confrontos das oitavas de final	Exibe os times classificados para as oitavas de final e seus respectivos confrontos
Exibição de estatísticas	Exibe ao usuário, as seguintes estatísticas: • A média geral de gols por jogo na 1a fase; • A média de gols por jogo em cada grupo; • Qual seleção fez mais gols em uma partida, informando quando foi o jogo, o time adversário e o placar.

Nas proximas subseções, é apresentado como se decorreu o desenvolvimento do código fonte para a atender a resolução do problema proposto.

2.2 Desenvolvimento

Para a manipulação dos dados do sistema foram utilizadas variáveis, condicionais do tipo *if* (se) e laços de repetição do tipo *while*(enquanto) e, principalmente, funções e listas.

Com a finalidade de melhor compreensão do código, é importante dividi-lo em subseções para melhor elucidação do que foi realizado em cada parte. Na seguintes subseções são apresentados e explicitados: as funções e o menu principal.

2.2.1 Funções

De modo a fazer a modularização do código, foram utilizadas funções que, segundo a documentação do Python, são subprogramas responsáveis por realizar processamentos menores no código e evitar repetições desnecessárias, desse modo seus resultados são recombinados em um programa principal.

No código do sistema ao total foram utilizadas 13 funções, onde são processados uma ou mais dados.

```
def menu_title(text):…

def is_number(text):…

def check_all_options(aux1, aux2):…

def check_menu_option():…

def check_submenu_option(menu_num):…

def group_option():…

def all_teams(n, list_all_teams):…

def group_letter(num_group):…

def create_groups(all_groups, list_all_teams, list_A, list_B, list_C, list_D, list_E, list_F, list_G, list_H):…

def show_all_groups():…

def matches(all_groups):…

def show_matches():…

def main():…
```

Figura 2. As funções que utilizadas durante a execução do código

A função *menu_title* é responsável por formatar os títulos das seções, ela recebe o texto com um parâmetro. A função pode ser observada na figura 3.

```
def menu_title(text):
    print('\n','==='*15)
    print(text)
    print('\n','==='*15)
```

Figura 3. A função *menu_title*

A função is_number é responsável por validar as entradas que devem ser somente de números. ela é utilizada para validar as entradas para os gols das seleções.a função pode ser observada na figura 4.

```
def is_number(text):
    n = ''
    while not n.isdigit():
        n = input(text)
        if not n.isdigit(): print('Valor inválido. Por favor, tente novamente!')
```

Figura 4. A função is_number

A função *check_all_options* é responsável por validar todas escolhas do menu e submenus, e dos grupos através de um laço de repetição *while* com o condicional *try/except*, que verifica se a entrada do usuário é um número inteiro e se está no intervalo das escolhas possíveis (aux1 - aux2) que são recebidos com parâmetros, isto se repete até que seja inserida uma entrada válida. ela retorna o valor escolhido pelo usuário para que prossiga para a execução do que o usuário escolheu, A função pode ser observada na figura 5.

```
def check_all_options(aux1, aux2):
    while True:
        n = input('\nPor favor, escolha uma das opções válidas: ')
        try:
            n = int(n)
            if aux1 <= n <= aux2:
                break
            else:
                print('\nOpção inválida, tente novamente!')
        except ValueError:
            print("\nSomente números são permitidos.")
    return n
```

Figura 5. A função *check_all_options* mostra o uso do try/except para validar a entrada

A função *check_menu_option* é usada para exibir as principais entradas do sistema com a entrada sendo validada pela função *check_all_options* (figura 5) no intervalo de (0-3). com *menu_num* == 1 para o submenu de cadastro, *menu_num* == 2 para o submenu de edição, e *menu_num* == 3 para o submenu de exibição e *menu_num* == 0 para encerrar o programa. a função pode ser observada na figura 6.

```
def check_menu_option():
    print('''\nDigite apenas o número da opção que deseja executar.\n
[1] - Cadastro \n[2] - Edição \n[3] - Exibição \n[0] - Sair do Cadastro
    menu_num = check_all_options(0,3)
    return menu_num
```

Figura 6. A função *check_menu_option* recebe e valida a entrada do usuário usuário

A função *check_submenu_option* funciona de modo semelhante a *check_menu_option*. ela receber a opção do menu como parâmetro e, exibe e faz validação das entradas do usuário para os submenus das três opções presentes no menu(figura 6). a função pode ser observada na figura 7.

```
def check_submenu_option(menu_num):
    print('\nO que deseja fazer agora?\n')
    if menu_num == 1:
        print('''[1] - Cadastrar seleções dos grupos \n[2] - Cadastrar confrontos \n[0] - Voltar ao menu anterior \n
        sub_menu_num = check_all_options(0,2)
    elif menu_num == 2:
        print('''[1] - Editar seleções e grupos \n[2] - Editar confrontos \n[0] - Voltar ao menu anterior \n=======
        sub_menu_num = check_all_options(0,2)
    elif menu_num == 3:
        print('''[1] - Exibir seleções e grupos \n[2] - Exibir confrontos por grupo \n[0] - Voltar ao menu anterior
        sub_menu_num = check_all_options(0,2)
    return sub_menu_num
```

Figura 7. A função *check_submenu_option*

A função *group_option* é responsável por mostrar ao usuário os grupos das seleções e validar a sua escolha. A função pode ser observada na figura 8.

```
def group_option():
    print('''\nQual Grupo deseja escolher?\n
[1] - Grupo A \n[2] - Grupo B \n[3] - Grupo C \n[4] - Grupo D \n[5] - Grupo E
[6] - Grupo F \n[7] - Grupo G \n[8] - Grupo H \n[0] - Voltar ao menu anterior
\n==================================================\n''')
    group_num = (check_all_options(0,8))
    return group_num
```

Figura 8. A função *group_option*

A função *all_teams* é responsável por receber a entrada de cada seleção e por validar, se, caso a seleção já esteja cadastrada em algum grupo, o usuário deve informar uma nova seleção. a função pode ser observada na figura 9.

```
def all_teams(n, list_all_teams):
    team = input(f'Digite a {n}ª seleção: ').upper()
    if team in list_all_teams:
        print('SELEÇÃO JÁ CADASTRADA. \nPOR FAVOR, ESCOLHA OUTRA!')
        team = input(f'Digite a {n}ª seleção: ').upper()
        while team in list_all_teams:
            team = input(f'Digite a {n}ª seleção: ').upper()
    return team
```

Figura 9. A função *all_teams* recebe o nome da seleção e checa se ela já está cadastrada no sistema.

```python
def group_letter(num_group):
    if num_group == 1:
        return 'A'
    elif num_group == 2:
        return 'B'
    elif num_group == 3:
        return 'C'
    elif num_group == 4:
        return 'D'
    elif num_group == 5:
        return 'E'
    elif num_group == 6:
        return 'F'
    elif num_group == 7:
        return 'G'
    elif num_group == 8:
        return 'H'
```

Figura 10. A função *group_letter*

A função *create_groups* é responsável por criar e alterar os grupos das seleções. inicialmente ela lê e receber as seleções do arquivo *groups.txt* para que possam ser alteradas. as seleções são alteradas conforme o grupo escolhido, mantendo as demais, dos outros grupos inalteradas. isso pode ser observado na figura 11a.

```python
def create_groups(list_groups, all_groups, list_all_teams, list
    num_group = group_option()
    #recuperando as seleções cadastradas no arquivo
    with open('groups.txt','r') as f:
        alist = [line.rstrip() for line in f]
        list_A = [alist[0], alist[1], alist[2], alist[3]]
        list_B = [alist[4], alist[5], alist[6], alist[7]]
        list_C = [alist[8], alist[9], alist[10], alist[11]]
        list_D = [alist[12], alist[13], alist[14], alist[15]]
        list_E = [alist[16], alist[17], alist[18], alist[19]]
        list_F = [alist[20], alist[21], alist[22], alist[23]]
        list_G = [alist[24], alist[25], alist[26], alist[27]]
        list_H = [alist[28], alist[29], alist[30], alist[31]]
```

Figura 11a. A função *create_groups* recebendo as seleções salvas no arquivo *groups.txt*

Seguidamente, a função recebe os nomes das seleções e salva na lista do grupo que foi escolhido. Essa parte está disposta na figura 11b.

```
Letter = group_letter(num_group)
menu_title(f'\nCADASTRO DO GRUPO {Letter}')

t1 = all_teams(1, list_all_teams)
list_all_teams.append(t1)

t2 = all_teams(2, list_all_teams)
list_all_teams.append(t2)

t3 = all_teams(3, list_all_teams)
list_all_teams.append(t3)

t4 = all_teams(4, list_all_teams)
list_all_teams.append(t4)

if num_group == 1:
    list_A = [t1, t2, t3, t4]
elif num_group == 2:
    list_B = [t1, t2, t3, t4]
elif num_group == 3:
    list_C = [t1, t2, t3, t4]
elif num_group == 4:
    list_D = [t1, t2, t3, t4]
elif num_group == 5:
    list_E = [t1, t2, t3, t4]
elif num_group == 6:
    list_F = [t1, t2, t3, t4]
elif num_group == 7:
    list_G = [t1, t2, t3, t4]
elif num_group == 8:
    list_H = [t1, t2, t3, t4]
```

Figura 11b. A função *create_groups* recebendo as seleções e salvando na lista conforme o grupo escolhido

Por fim, a função salva as seleções no arquivo groups.txt. Esse processo é apresentado na figura 11c.

```
with open('groups.txt','w+') as arq:
    arq.write('\n'.join(list_A))
    arq.write('\n')
    arq.write('\n'.join(list_B))
    arq.write('\n')
    arq.write('\n'.join(list_C))
    arq.write('\n')
    arq.write('\n'.join(list_D))
    arq.write('\n')
    arq.write('\n'.join(list_E))
    arq.write('\n')
    arq.write('\n'.join(list_F))
    arq.write('\n')
    arq.write('\n'.join(list_G))
    arq.write('\n')
    arq.write('\n'.join(list_H))
arq.close()
```

Figura 11c. A função *create_groups* salvando as seleções no arquivo conforme *groups.txt*

A função *show_all_groups* é responsável por exibir aos grupos e suas respectivas seleções, ela lê o arquivo *groups.txt* e exibe as seleções cadastradas. os grupos só são exibidos caso todas as seleções estejam cadastradas. Ela pode ser observada na figura 12.

```
def show_all_groups():
    with open('groups.txt','r') as f:
        alist = [line.rstrip() for line in f]
    if len(alist) >= 32:
        menu_title('            GRUPOS CADASTRADOS')
        menu_title('           Grupo A')
        print(alist[0],'|',  alist[1],'|',  alist[2],'|',  alist[3])
        menu_title('           Grupo B')
        print(alist[4],'|', alist[5],'|',  alist[6],'|',  alist[7])
        menu_title('           Grupo C')
        print(alist[0],'|',  alist[9],'|',  alist[10],'|',  alist[11])
        menu_title('           Grupo D')
        print(alist[12],'|',  alist[13],'|',  alist[14],'|',  alist[15])
        menu_title('           Grupo E')
        print(alist[16],'|',  alist[17],'|',  alist[18],'|',  alist[19])
        menu_title('           Grupo F')
        print(alist[20],'|',  alist[21],'|',  alist[22],'|',  alist[23])
        menu_title('           Grupo G')
        print(alist[24],'|',  alist[25],'|',  alist[26],'|',  alist[27])
        menu_title('           Grupo H')
        print(alist[28],'|',  alist[29],'|',  alist[30],'|',  alist[31])
    else:
        print('\nFALTAM GRUPOS A SEREM CADASTRADOS\n')
```

Figura 12. A função *show_all_groups*

A função *matches* é responsável por criar e alterar os confrontos dos grupos. inicialmente ela lê e receber as seleções do arquivo *groups.txt* para que possam ser utilizadas. as seleções são alteradas conforme o grupo escolhido, mantendo as demais, dos outros grupos inalteradas. isso pode ser observado na figura 13a.

```
def matches(all_groups):
    num_group = group_option()
    #recuperando as seleções do arquivo de grupos
    with open('groups.txt','r') as f:
        alist = [line.rstrip() for line in f]
        list_A = [alist[0], alist[1], alist[2], alist[3]]
        list_B = [alist[4], alist[5], alist[6], alist[7]]
        list_C = [alist[8], alist[9], alist[10], alist[11]]
        list_D = [alist[12], alist[13], alist[14], alist[15]]
        list_E = [alist[16], alist[17], alist[18], alist[19]]
        list_F = [alist[20], alist[21], alist[22], alist[23]]
        list_G = [alist[24], alist[25], alist[26], alist[27]]
        list_H = [alist[28], alist[29], alist[30], alist[31]]
```

Figura 13a. recebendo os grupos das seleções do arquivo txt

Seguidamente, a função indexa as seleções dos grupos para que possam ser utilizadas no cadastro dos confrontos, além de recuperar, do arquivo matches.txt, os confrontos já cadastrados, para que possam, ou não, serem atualizados.

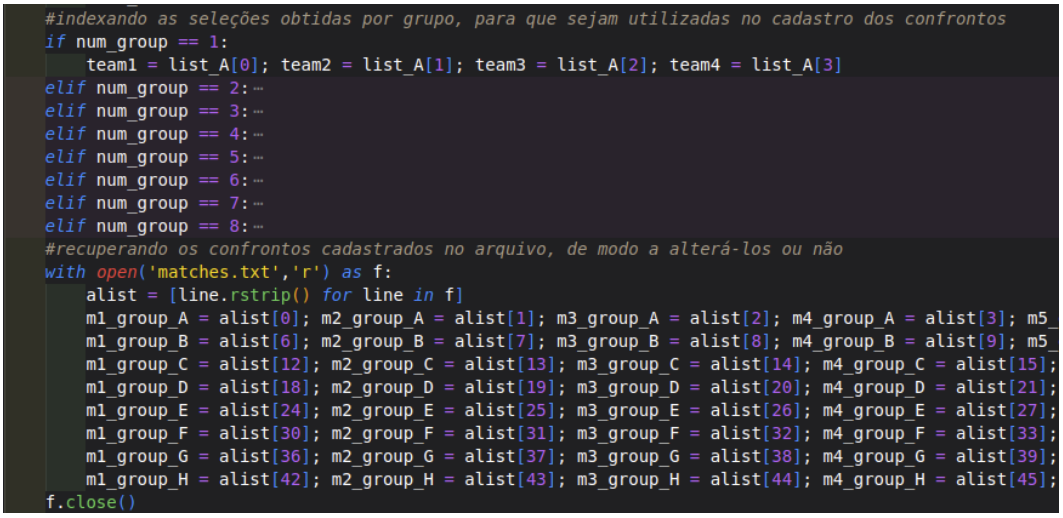

Figura 13b. anexando as seleções nas variáveis times e recuperando os confrontos já cadastrados no arquivo.

Por fim, a função recebe as entradas para os 6 confrontos e os salva, conforme o grupo escolhido, de modo a atualizar os dados e manter os demais confrontos no arquivo. Isto é mostrado nas figuras 13c e 13d.

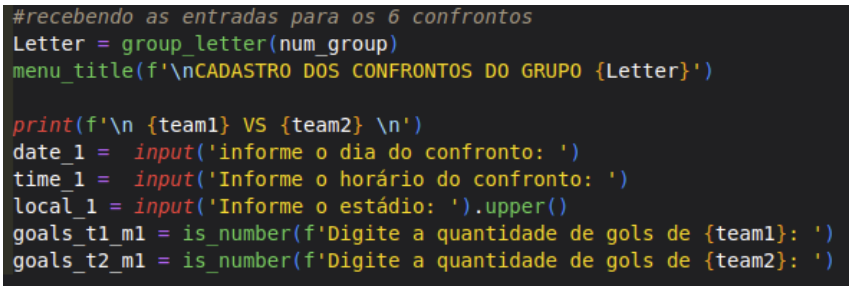

Figura 13c. variáveis dos dados do confronto 1, processo semelhante acontece para os demais confrontos .

Figura 13d. atualizando os confrontos conforme o grupo escolhido e, salvando todos os 48 confrontos no arquivo *matches.txt* .

A função *show_matches* é responsável por mostrar os confrontos de cada grupo, conforme a solicitação do usuário. Ela é observada na figura 14.

```
def show_matches():
    num_group = group_option()
    with open('matches.txt') as f:
        alist = [line.rstrip() for line in f]
        menu_title('            CONFRONTOS DOS GRUPOS CADASTRADOS')
        if num_group == 1:
            menu_title('           Grupo A')
            print(alist[0])
            print(alist[1])
            print(alist[2])
            print(alist[3])
            print(alist[4])
            print(alist[5])
```

Figura 14. A função *show_matches*

2.2.2 Menu inicial

No menu inicial, que é disposto dentro da função *main*, são inicializadas algumas listas e é solicitada a primeira entrada ao usuário através da função *check_menu_option* (figura 6). Essa entrada permite ao jogador escolher o que ele fará em seguida. O menu principal é mostrado na figura 15.

```
    list_all_teams = []; list_A = []; list_B = []
    list_C = []; list_D = []; list_E = []; list_F = []; list_G = []; list_H = []
    print('''================================================================================
        BEM-VINDOS AO SISTEMA DE CADASTRO DE JOGOS DA COPA DO MUNDO DE 2022
================================================================================''')
    num_menu = check_menu_option()
```

Figura 15. Menu Inicial.

A variável *num_menu* recebe o resultado da função *check_menu_option* (figura 6) e é utilizada na verificação de um laço de repetição *while (num_menu != 0)* onde o valor digitado é processado por condicionais *if* (se) onde no primeiro com (*num_menu == 1*) são dispostas as instruções para a cadastro, no segundo com (*num_menu == 2*) são dispostas as instruções para edição e no terceiro com (*num_menu == 3*), são dispostas as instruções para exibição. a figura 16 mostra essas instruções.

```
sub_menu_num = check_submenu_option(num_menu)
while sub_menu_num != 0:

    #menu responsável pela cadastro dos grupos e seleções e dos confrontos de cada grupo.
    if num_menu == 1:
        if sub_menu_num == 1: create_groups(list_all_teams, list_A, list_B, list_C, list_D
        elif sub_menu_num == 2: matches()
        sub_menu_num = check_submenu_option(num_menu)

    #menu responsável pela edição dos grupos e seleções.
    elif num_menu == 2:
        if sub_menu_num == 1: create_groups(list_all_teams, list_A, list_B, list_C, list_D
        elif sub_menu_num == 2: matches()
        sub_menu_num = check_submenu_option(num_menu)

    #menu responsável pela exibição dos grupos e seleções.
    elif num_menu == 3:
        if sub_menu_num == 1: show_all_groups()
        elif sub_menu_num == 2: show_matches()
        sub_menu_num = check_submenu_option(num_menu)

num_menu = check_menu_option()
```

Figura 16. Instruções presentes no menu inicial.

2.2.3 Saída do sistema

Quando o usuário escolhe a opção zero no menu inicial, as demais partes do código não são mais processadas e é exibida a mensagem disposta abaixo antes da execução do sistema ser interrompida.

```
#mensagem de encerramento do sistema.
time.sleep(1)
menu_title('\n      CADASTRO ENCERRADO COM SUCESSO!!!')
```

Figura 17. Mensagem de encerramento do sistema

Na seguinte seção são apresentados os resultados obtidos com a conclusão do problema, um manual de uso para o usuário e os erros que acontecem durante a execução.

3. Resultados e discussões

O código tem como objetivo principal ser um sistema de cadastramento da copa do mundo de futebol de 2022, com a inserção de seleções e confrontos. O sistema foi construído de maneira que sua utilização fosse a mais simples possível, facilitando a experiência do usuário.

A seguir são apresentados os passo-a-passos que o usuário deve seguir para utilizar o sistema de cadastramento.

3.1 Manual de Uso

Ao executar o sistema, o usuário encontra o menu inicial conforme foi detalhado na figura 15. Na figura 18, ele é mostrado sob a visão do usuário.

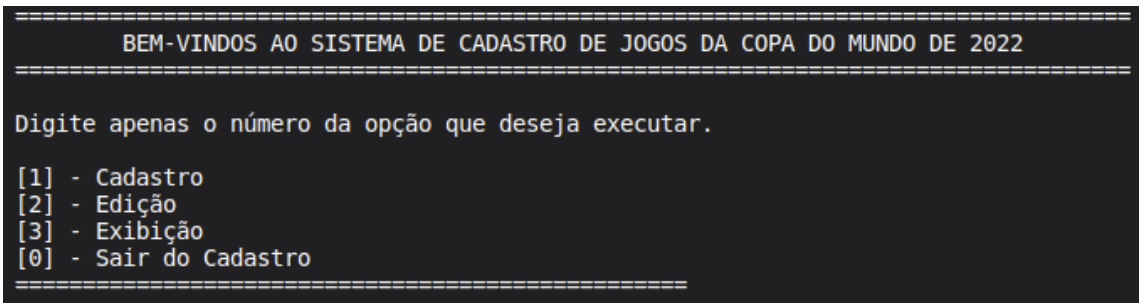

Figura 18. Menu inicial do sistema

3.1.1 Cadastro de um grupo

Conforme explicitado na sub subseção 2.2.2, ao escolher a opção [1] é apresentado o submenu de cadastro.

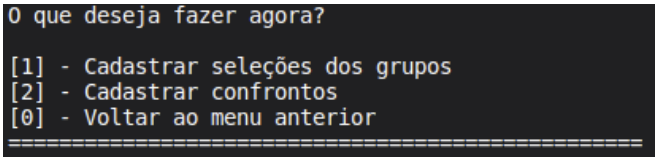

Figura 19. submenu de cadastramento

Subsequentemente, caso o usuário escolha cadastrar seleções, são apresentadas as escolhas de grupos (figura 8). A visão do usuário é apresentada na figura 20.

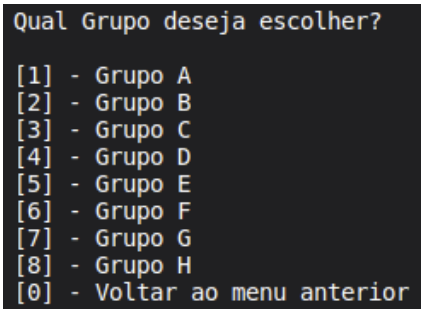

Figura 20. menu de entrada de grupos

Após a escolha do grupo, o usuário deve informar as quatro seleções do grupo para realizar o cadastro. esse cadastro é mostrado na figura 21.

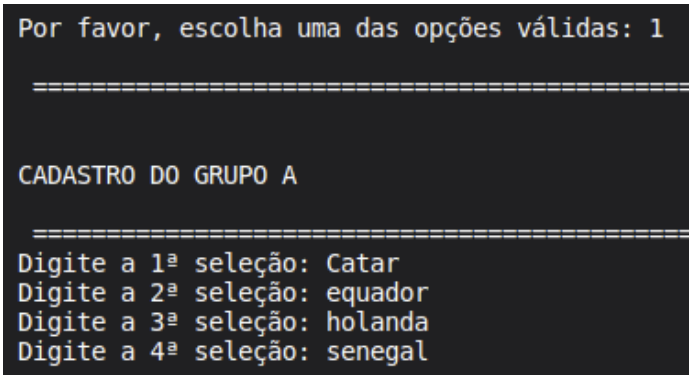

Figura 21. Cadastro das seleções de um grupo

Em seguida, o sistema reapresenta o submenu, para que o usuário escolha o que fazer em seguida. Caso seja escolhida a opção [2] do submenu, o sistema inicia o cadastro de confrontos, que novamente solicita o grupo (figura 20). Logo após, são apresentadas as entradas dos seis confrontos que devem ser preenchidos pelo usuário.

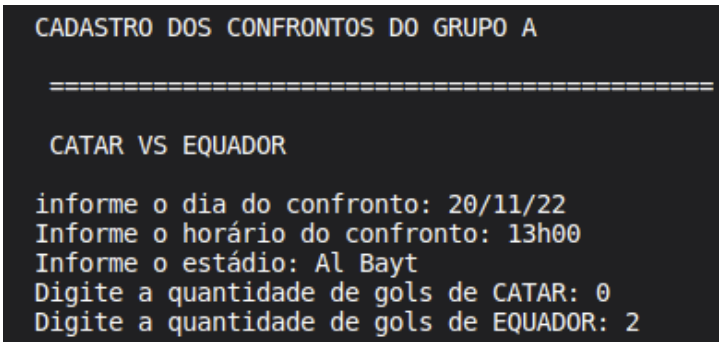

Figura 22. entradas do confronto de grupo

Caso o usuário opte por editar grupos e confrontos (*menu_option == 2*), o usuário é chamado a realizar novamente ou o cadastro do grupo ou das seleções de um grupo.

Se o usuário escolher a opção 3 no menu inicial, é exibido o submenu de exibição. Este submenu está disposto na figura 23.

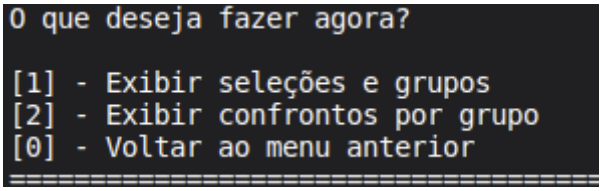

Figura 23. submenu de exibição

Caso o usuário escolha a opção [1], são exibidos os oito grupos com suas respectivas seleções, conforme a figura 24.

Figura 24. alguns dos grupos sendo exibidos

Logo, caso o usuário escolha a opção [2], são exibidos os confrontos de cada grupo, conforme solicitação.

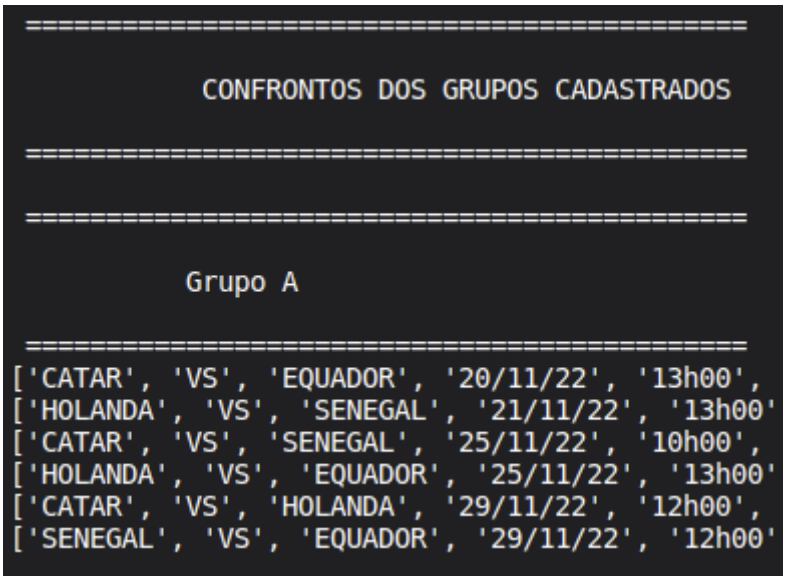

Figura 25. confrontos do grupo A sendo exibidos

Por fim, caso o usuário escolha as opções [0] em todos os menus e submenus, o sistema volta para o menu anterior, e no caso do menu principal, o sistema é encerrado. A mensagem de encerramento está disposta na figura 26.

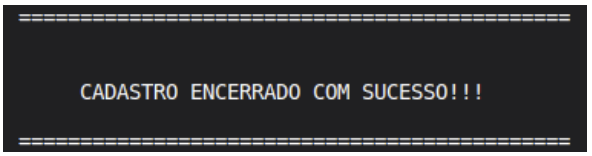

Figura 26. mensagem de encerramento do jogo

Na seguinte subseção, são apresentados os problemas encontrados durante a execução do sistema.

3.2 Bugs

Após o desenvolvimento de todo o código fonte foram constatadas falhas. a primeira é que o sistema permite a entrada de seleções iguais, caso as seleções sejam cadastradas em momentos distintos, e a segunda é que o sistema permite a entrada de números na entrada das seleções. Demais erros não foram encontrados após a conclusão e em testes do código fonte.

Na seguinte seção é realizada a conclusão do relatório e as considerações finais.

4. Conclusão

Com a conclusão do sistema pode-se colocar em prática o uso de algumas das estruturas básicas do python sendo elas as funções, listas, estruturas condicionais e laços de repetição. Os requisitos de entrada solicitados foram atendidos.

Contudo, para os requisitos de saída, apenas o cadastro e exibição dos grupos e seus confrontos são atendidos, de modo que para editar, se faz necessário realizar um novo cadastramento.

Referências

PYTHON SOFTWARE FOUNDATION. Entrada e saída. 2022. Disponível em: https://docs.python.org/pt-br/3/tutorial/inputoutput.html. Acesso em: 13 set. 2022

PYTHON SOFTWARE FOUNDATION. Funções. 2022. Disponível em: https://docs.python.org/pt-br/3.10/reference/compound_stmts.html. Acesso em: 02 nov. 2022

PYTHON SOFTWARE FOUNDATION. Estrutura de Dados. 2022. Disponível em: https://docs.python.org/pt-br/3/tutorial/datastructures.html. Acesso em: 02 nov. 2022

PYTHON SOFTWARE FOUNDATION. Funções Embutidas. 2022. Disponível em: https://docs.python.org/pt-br/3/library/functions.html. Acesso em: 02 nov. 2022

MIRANDA, Taynná. Copa do Mundo: uma paixão tipicamente brasileira .2022 Disponível em: https://ludopedio.org.br/arquibancada/copa-do-mundo-uma-paixao-tipicamente-brasileira/. Acesso em: 10 dez. 2022