Relatório do Problema 3 – Copa do Mundo 2022

Lucas Gabriel Cerqueira Santos Lima

Departamento de Ciências e Exatas – Universidade Estadual de Feira de Santana (UEFS)

Feira de Santana, Bahia – Brasil

lucasfacul20222@gmail.com

Resumo. *Visando auxiliar ao acompanhamento detalhado da Copa do Mundo 2022, foi proposto o desenvolvimento de um software capaz de receber informações sobre o evento e fornecer seus dados e estatísticas. O objetivo consiste em formular um software funcional e modularizado, que garanta a integridade dos dados e consistência das informações lançadas através de registros.*

Abstract. *Aiming to assist in the detailed monitoring of the 2022 World Cup, the development of software capable of receiving information about the event and providing its data and statistics was standard. The objective is to formulate a functional and modularized software, which guarantees the integrity of the data and consistency of the information released through records.*

1. Introdução

A Copa do Mundo é uma competição de futebol realizada pela Federação Internacional de Futebol a cada quatro anos. Esse evento é a maior e mais importante competição de futebol organizada pela Fifa e, em sua estrutura atual, é composto por 32 seleções. Os jogos acontecem em uma nação-sede determinada por meio de eleição estruturada pela própria Fifa.

A importância desse evento é claramente representada com base em dados disponibilizados pela própria Fifa. Eles mostram que a Copa de 2014, realizada no Brasil, e a de 2010, na África do Sul, contaram com cerca de 3,2 bilhões de espectadores. No caso da Copa de 2014, o jogo final, disputado entre Alemanha e Argentina, foi assistido por 1 bilhão de pessoas. Ademais, a Copa do Mundo também fomenta a paixão mundial pelo esporte mais popular: o futebol. Tamanha é sua relevância, que alguns países decretam feriados no dia dos jogos de sua seleção representante.

Com a globalização, a tecnologia tornou-se uma ferramenta indispensável no dia a dia, modernizando e facilitando o acesso aos meios de comunicação e consequentemente popularizando ainda mais esse evento tão grandioso. Dessa forma, os espectadores e amantes da Copa do Mundo, procuram cada vez mais formas de acompanhar e se inteirar sobre os acontecimentos do evento.

Como resultado, levando em consideração a relevância da Copa do Mundo globalmente, os alunos de Engenharia de Computação da UEFS, a pedido de um experiente ex-jogador de um grandioso clube, desenvolveram um software capaz de

cadastrar as seleções participantes do evento em seus determinados grupos e seus respectivos jogos com informações como: data, horário, local e placar. Todas essas informações são preservadas, permitindo alterações, exclusões e acompanhamento das estatísticas.

A seguir, será descrito no tópico “Metodologia”, as escolhas para a solução do problema e as ferramentas necessárias para o uso do programa. Na seção de “Resultados e Discussões”, será exposto os passos detalhados para uso do programa. Finalizando, será apresentado em “Conclusão”, se os objetivos propostos foram cumpridos e as possíveis melhorias para o programa.

2. Metodologia

Com o objetivo de desenvolver o software em questão, foram realizadas sessões PBL. Após discussões, decidiu-se desenvolver o código-fonte do aplicativo de acompanhamento da Copa do Mundo 2022 na linguagem de programação Python. Foi estabelecido que o código deveria ser modularizado adequadamente, garantindo a integridade dos dados em arquivos e que permitisse alterações e exclusões dos mesmos.

Durante as sessões, sugeriu-se que fossem utilizadas estruturas como listas, dicionários, verificações de entrada de dados, utilização de funções, para a organização e armazenamento em arquivos das informações necessárias para o funcionamento do programa. O programa deve permitir que seja cadastrado as quatro equipes dos oito grupos, além de seus respetivos confrontos e quando solicitado, caso todas a informações já tenham sido lançadas no sistema, exibir as estatísticas requisitadas.

Todo o programa foi desenvolvido para o Sistema Operacional Windows, em linguagem Python na versão 3.8.10. A IDE utilizada foi a Visual Studio Code na versão 1.70.3 no sistema operacional Microsoft Windows 7 Ultimate.

2.1. Definição de requisitos

Foram definidos os requisitos para o software de acompanhamento da Copa do Mundo 2022. Os requisitos foram divididos em três tabelas. A Tabela 1 contém os requisitos do sistema para realizar o cadastro dos grupos e jogos.

Tabela 1. Tabela 1. Requisitos do software para cadastro

Grupos	O usuário deve escolher um dos oito grupos (A à H) para cadastrar as seleções. Grupos já cadastrados ou seleções cadastradas em outros grupos não podem ser cadastrados novamente.
Jogos	O usuário deve escolher um dos oito grupos (A à H), escolher um dos seis jogos disponíveis, e logo após cadastrar suas informações. Grupos não cadastrados não podem cadastrar jogos. Jogos já

	cadastrados não podem ser cadastrados novamente.

Na Tabela 2, foram organizadas as funcionalidades para edição dos grupos ou jogos.

Tabela 2. Funcionalidades para edição

Grupos	Caso deseje mudar uma seleção por outra seleção que já esteja cadastrada em algum grupo, existe a possibilidade de trocar essas seleções de grupos (uma vai para o outro) ou cancelar a operação. Caso mude uma seleção, todos os confrontos da mesma precisarão ser reescritos.
Jogos	Caso deseje editar algum jogo, o usuário deve escolher o grupo e o determinado jogo, no qual ele pode escolher qual dado em específico alterar (data, placar, estádio, horário ou cartões) ou excluir todos os dados deste jogo.

Na tabela 3, foram organizadas as informações que deverão conter no relatório final da fase de grupos.

Tabela 3. Informações exibidas no relatório

Confrontos das oitavas	Exibe os confrontos diretos entre as seleções classificadas em primeiro e segundo de cada grupo.
Média de gols	Exibe as estatísticas sobre as médias de gols por grupo e a média total de gols da primeira fase.
Maior goleada	Exibe as informações da partida em que houve o maior número de gols de uma equipe.
Todas as informações	Exibe todas as informações detalhadas acima.

Contém ainda, a opção de visualizar os grupos e os jogos de algum grupo que já foi cadastrado, podendo ter acesso a quais ainda faltam cadastrar. Também foi determinado que ao encerrar o programa, os dados recebidos deveriam ser armazenados em arquivos através de módulos como json, texto, binário ou outros.

2.2. Descrição do algoritmo

O software foi desenvolvido criando um menu funcional onde o usuário pode escolher entre cadastrar um grupo ou jogo de um grupo, editar as informações de algum grupo ou jogos de um grupo, visualizar os grupos e jogos cadastrados, exibir o relatório final e sair do programa.

2.2.1. Funções utilizadas

Para modularização do programa foram utilizadas funções. Elas são compostas por vários comandos que ajudam na otimização do código e facilitam a alteração de algum trecho do mesmo. A seguir será mostrado as principais funções que foram implementadas.

2.2.1.1. Funções para cadastro dos grupos e jogos

Para armazenar o cadastro dos grupos e jogos na memória, foram utilizados dois dicionários onde cada um armazena as seleções dos grupos A à H e o outro armazena os confrontos dos jogos de cada grupo. Visando facilitar o cadastro desses dados, foram criadas as funções *cadastrar_grupos* e *cadastrar_jogos*. A primeira função citada, verifica se o grupo escolhido já possui cadastro, caso não possua, chama a função *inserir_seleção* responsável por verificar se a seleção desejada já possui cadastro em outro grupo, caso não possua, a seleção é adicionada no dicionário dos grupos, conforme Imagem 1.

```python
#Função para cadastrar os grupos
def cadastrar_grupos(grupos, jogos):
    letra = pegar_letra(grupos)
    if len(grupos[letra]) >0:
        print("Este grupo já está cadastrado!\n")
    else:
        while len(grupos[letra]) < 4:
            limpar_cmd()
            inserir_selecao(grupos, letra)
        atualizar_partidas(grupos[letra], letra, jogos)
        print("Cadastro realizado!")
    print("Retornando ao menu inicial...")
    limpar_cmd(True)

#Função para inserir as seleções no grupo escolhido
def inserir_selecao(grupos, letra):
    equipe = pegar_selecao(f"Digite o nome da seleção:\n>>> ")
    indisponivel, grupo = verificar_selecao(equipe, grupos)
    if indisponivel:
        print(f"Esta seleção ja está no grupo {grupo}!")
        limpar_cmd(True)
        inserir_selecao(grupos, letra)
    else:
        grupos[letra].append(equipe)
        limpar_cmd()
```

Imagem 1. Funções para cadastro dos grupos

A função *cadastrar_jogos* verifica todos os jogos já cadastrados e remove os que já foram, limtando ao usuário escolher apenas os que ainda não foram. Ao escolher o jogo disponível, o usuário pode dar entrada nas informações do mesmo, conforme Imagem 2.

```python
#Função para cadastrar os jogos de um grupo
def cadastrar_jogos(grupos, jogos):
    letra = pegar_letra(grupos)
    if letra in jogos.keys():
        para_deletar = []
        partidas = list(jogos[letra].keys())
        for p in partidas:
            if len(jogos[letra][p]) > 2:
                para_deletar.append(p)
        for p in para_deletar:
            partidas.remove(p)
        if len(partidas) == 0:
            print("Todos os jogos deste grupo já foram cadastrados!\n")
            print("Retornando ao menu inicial...")
            limpar_cmd(True)
            return True
        indice_partida = menu("ESCOLHA A PARTIDA", *partidas)
        partida = partidas[indice_partida]
        jogos[letra][partida]['estadio'] = input("Digite o nome do estádio:\n>>> ").title()
        limpar_cmd()
        jogos[letra][partida]['data'] = pegar_data()
        jogos[letra][partida]['horario'] = pegar_hora()
        lancar_resultados(grupos, jogos, letra, partida)
        return False
    else:
        return True
```

Imagem 2. Funções para cadastro dos jogos

2.2.1.2. Funções para edições dos grupos e jogos

Para a edição dos grupos foi criada a função *editar_grupos*. Ela verifica se o grupo já possui as quatro seleções cadastradas, caso já possua, possibilita a escolha de excluir todas as seleções do grupo ou alterar alguma das quatro seleções. Para alterar alguma seleção, a função primeiro procura se a nova seleção já está em algum grupo, caso já esteja, possibilita a opção de mandar a antiga seleção para o grupo na qual a nova seleção está e a nova seleção é cadastrada no grupo. Caso a nova seleção desejada não possua cadastro em outro grupo, ela substitui o lugar da antiga, conforme Imagem 3.

```python
#Função para editar um grupo
def editar_grupos(grupos, jogos):
    letra = pegar_letra(grupos)
    if len(grupos[letra]) == 0:
        print("Este grupo ainda não foi cadastrado!")
    else:
        acao = menu("ESCOLHA UMA AÇÃO", "Limpar Grupo", "Trocar uma Seleção", "Voltar")
        if acao == 0:
            grupos[letra] = []
            atualizar_partidas(grupos[letra], letra, jogos)
        elif acao == 1:
            indice_selecao = menu('ESCOLHA QUAL SELEÇÃO REMOVER', *grupos[letra],'Voltar')
            if indice_selecao!=4:
                selecao = grupos[letra][indice_selecao]
                nova_selecao = pegar_selecao("Digite o nome da nova seleção:\n>>> ").title()
                indisponivel, n_letra = verificar_selecao(nova_selecao, grupos)
                if indisponivel:
                    troca = menu(f"SELEÇÃO NO GRUPO {n_letra}!\n\nDeseja Trocar?", "Sim", "Não")
                    if troca == 0:
                        grupos[letra][indice_selecao] = nova_selecao
                        n_indice = grupos[n_letra].index(nova_selecao)
                        grupos[n_letra][n_indice] = selecao
                        atualizar_partidas(grupos[n_letra], n_letra, jogos)
                        atualizar_partidas(grupos[letra], letra, jogos)
                    else:
                        print("Cancelando operação...")
                        limpar_cmd(True)
                else:
                    grupos[letra][indice_selecao] = nova_selecao
                    atualizar_partidas(grupos[letra], letra, jogos)

    print("Retornando ao menu inicial...")
    limpar_cmd(True)
```

Imagem 3. Funções para edição dos grupos

Para a edição dos jogos foi desenvolvida a função *editar_jogos*. Ela primeiro verifica se o grupo escolhido para edição dos jogos já possui cadastro, caso já possua, libera a opção de escolher entre um dos seis jogos do grupo, conforme Imagem 4. Ademais, é verificado se o jogo escolhido já foi cadastrado, caso já tenha sido, o usuário tem a opção de escolher entre alterar algum dado do jogo ou excluir todos os dados do

jogo, conforme Imagem 5. Caso escolha excluir todos os dados, as informações gravadas no valor da chave do jogo são apagadas.

```python
#Função para editar um jogo
def editar_jogos(grupos, jogos):
    letra = pegar_letra(grupos)
    if len(grupos[letra]) == 0:
        print("Este grupo ainda não foi cadastrado!")
    else:
        partidas = list(jogos[letra].keys())
        indice_jogo = menu("ESCOLHA O JOGO", *partidas)
        jogo = partidas[indice_jogo]

        dados = list(jogos[letra][jogo].keys())
        dados.remove("equipe 1")
        dados.remove("equipe 2")
        if len(dados) == 0:
            print("Este jogo não tem dados!")
```

Imagem 4. Função verificar se o jogo possui cadastro

```python
    else:
        acao = menu("OQUE DESEJA FAZER", "Alterar dados", "Apagar dados", "Voltar")
        if acao == 0:
            escolha = menu("ESCOLHA QUAL DADO ALTERAR", *dados, 'Voltar')
            if escolha != 5:
                dado  = dados[escolha]
                if dado == "estadio":
                    n_informacao = input("Digite o nome do novo estádio:\n>>> ")
                    jogos[letra][jogo][dado] = n_informacao
                elif dado == "data":
                    n_informacao = pegar_data()
                    jogos[letra][jogo][dado] = n_informacao
                elif dado == "horario":
                    n_informacao = pegar_hora()
                    jogos[letra][jogo][dado] = n_informacao
                elif dado == "placar":
                    n_informacao = pegar_placar(jogos[letra][jogo]['equipe 1'], jogos[letra][jogo]['equipe 2'])
                    jogos[letra][jogo][dado] = n_informacao
                elif dado == "cartoes":
                    n_informacao = pegar_cartoes(jogos[letra][jogo]['equipe 1'], jogos[letra][jogo]['equipe 2'])
                    jogos[letra][jogo][dado] = n_informacao
        elif acao == 1:
            dado = jogos[letra][jogo]
            dado.pop('estadio')
            dado.pop('data')
            dado.pop('horario')
            dado.pop('cartoes')
            dado.pop('placar')
            jogos[letra][jogo] = dado
print("Retornando ao menu inicial...")
limpar_cmd(True)
```

Imagem 5. Função para edição dos jogos

2.2.1.3. Função para exibição dos grupos e jogos cadastrados

Para visualizar quais grupos e jogos já foram cadastrados, foi implementada a função *visualizacoes*. Esta função permite ao usuário escolher entre ter acesso aos grupos ou aos jogos cadastrados. Primeiramente, independente da escolha, a função verifica se possui alguma informação a ser mostrada, ou seja, se possui pelo menos um grupo ou um jogo cadastrado. Caso possua, ela irá exibir os grupos ou jogos que já foram cadastrados, conforme Imagem 6 e Imagem 7.

```python
#Função para mostrar os grupos ou os jogos cadastrados
def visualizacoes(grupos, jogos):
    vazio = 0
    escolha = menu("ESCOLHA OQUE VISUALIZAR", "Grupos", "Jogos", "Voltar")

    if escolha == 0:
        for chave in grupos:
            if len(grupos[chave]) > 0:
                corda()
                print(f"°Grupo {chave}\n")
                for time in grupos[chave]:
                    print(f"-> {time}")
                print()
            else:
                vazio +=1
        if vazio == 8:
            print("Ainda não há grupos cadastrados!")

    corda()
    input('Pressione ENTER para continuar...')
    limpar_cmd()
```

Imagem 6. Função para exibir os grupos cadastrados

```python
#Função para mostrar os grupos ou os jogos cadastrados
def visualizacoes(grupos, jogos):
    vazio = 0
    escolha = menu("ESCOLHA OQUE VISUALIZAR", "Grupos", "Jogos", "Voltar")

    if escolha == 1:
        if len(jogos) > 0:
            for grupo in jogos:
                n = 0
                corda()
                print(f"\n°Grupo {grupo}\n")
                for partida in jogos[grupo]:
                    n += 1

                    print(f'{n}. {partida} -> ', end  ="")
                    for dado in jogos[grupo][partida]:
                        if dado not in ['equipe 1', "equipe 2"]:
                            print(f'( {dado}: {jogos[grupo][partida][dado]} )', end = " | ")
                    print('\n')
        else:
            print("Ainda não há jogos cadastrados!")

    corda()
    input('Pressione ENTER para continuar...')
    limpar_cmd()
```

Imagem 7. Função para exibir os jogos cadastrados

2.2.1.4. Funções definir os confrontos das oitavas

Foi criada a função *classificar* para selecionar as seleções com maior "score" de cada grupo. Primeiro ela verifica qual das quatro seleções de um grupo fez a maior pontuação, verificando grupo por grupo até percorrer todo o dicionário. Caso haja um empate, ela verifica a seleção com maior saldo de gols. Caso haja um empate, ela verifica qual seleção marcou mais gols. Caso haja um empate, ela verifica qual seleção possui um menor número de cartões amarelos recebidos. Segue demonstração na Imagem 8.

```python
#Função para verificar qual seleção possui o maior "score"
def classificar(letra, grupo, jogos):
    selecao1 = grupo[0]
    for s in range(1, len(grupo)):
        s1_pontos = pontos_totais(letra, jogos, selecao1)
        s_pontos = pontos_totais(letra, jogos, grupo[s])

        if s1_pontos < s_pontos:  #classifica pela pontuação
            selecao1 = grupo[s]

        elif  s_pontos == s1_pontos:  #classifica pelo saldo de gols
            s1_gols = saldo_de_gols(jogos[letra], selecao1)
            s_gols = saldo_de_gols(jogos[letra], grupo[s])
            if s1_gols < s_gols:
                selecao1 = grupo[s]

            elif s1_gols == s_gols:   #classifica pelos gols marcados
                gols_s1 = gols_time(jogos[letra], selecao1)
                gols_s = gols_time(jogos[letra], grupo[s])
                if gols_s1 < gols_s:
                    selecao1 = grupo[s]

                elif gols_s1 == gols_s:   #classifica pela menor quantidade de cartões
                    cartoes_s1 = quantidade_cartoes(jogos[letra], selecao1)
                    cartoes_s = quantidade_cartoes(jogos[letra], grupo[s])
                    if cartoes_s1 < cartoes_s:
                        selecao1 = grupo[s]

    return selecao1
```

Imagem 8. Função para determinar seleção com maior “score”

Para selecionar o primeiro e segundo lugar do grupo, foi criada a função *classificados*. Ela primeiro determina a seleção que possui o maior “score”, levando em consideração todos os critérios de desempate. Após isso, ela remove essa seleção e faz novamente a comparação entre as seleções restantes. Das seleções que restaram, a que obtiver o maior “score” será determinada a segunda seleção classificada, conforme Imagem 9.

```python
#Função para pegar os classificados
def classificados(letra, grupo, jogos):
    selecoes = grupo.copy()
    primeiro = classificar(letra, selecoes, jogos)
    selecoes.remove(primeiro)
    segundo = classificar(letra, selecoes, jogos)

    return primeiro, segundo
```

Imagem 9. Função para determinar primeiro e segundo do grupo

Foi criada a função *jogos_das_oitavas* para definir o confronto das seleções classificadas. Ela conta com três listas: uma para armazenar os confrontos das seleções, uma para armazenar as seleções classificadas em primeiro lugar e uma para armazenar as seleções classificadas em segundo lugar. Após ser armazenadas todas as equipes classificadas, a função alterna os confrontos entre o primeiro lugar de um grupo e o segundo lugar do grupo em sequência e adiciona na lista dos confrontos, conforme Imagem 10.

```python
#Função que define os confrontos das oitavas
def jogos_das_oitavas(grupos, jogos):
    partidas = []
    times1 = []
    times2 = []

    for grupo in grupos: #separa as duplas
        time1, time2 = classificados(grupo, grupos[grupo], jogos)
        times1.append(time1)
        times2.append(time2)

    #Alternando entre as duplas
    for i in range(len(times1)):
        partidas.append(f"{times1[i]} x {times2[i+1]}" if i %2 == 0 else f"{times1[i]} x {times2[i-1]}")

    return partidas
```

Imagem 10. Função para determinar os confrontos das oitavas

2.2.1.5. Funções para exibição do relatório final

Primeiramente foi criada a função *liberar_relatorio* para verificar se todos os grupos e todos os jogos dos grupos já foram cadastrados. Ela verifica se em algum jogo não possui a informação relacionada ao placar da partida, caso possua esta informação em todos os jogos cadastrados, o relatório é liberado, conforme Imagem 11.

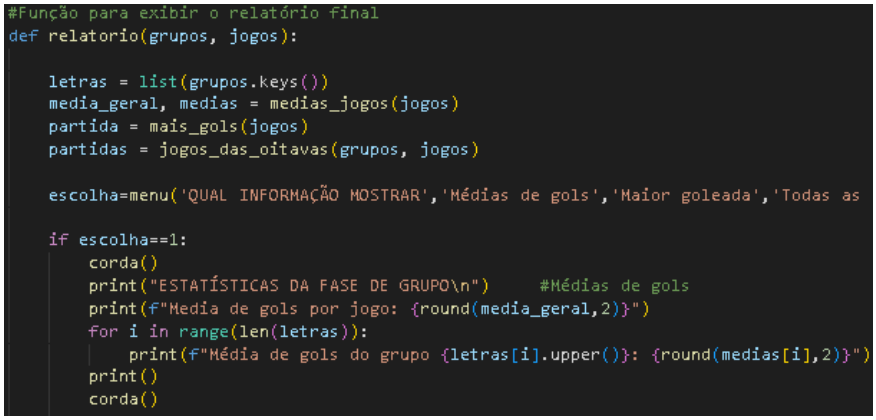
```
#Função para verificar se todos os resultados ja foram lançados
def liberar_relatorio(jogos):
    for grupo in jogos:
        for partidas in jogos[grupo]:
            if 'placar' not in jogos[grupo][partidas].keys():
                return False
    return True if len(jogos) > 0 else False
```

Imagem 11. Função para verificar liberação do relatório final

Para exibição do relatório final foi criada a função *relatório*, que recebe todas as informações necessárias das funções criadas anteriormente (média de gols, confronto das oitavas e etc). O usuário tem a opção de visualizar cada dado individualmente ou ter acesso a exibição de todos os dados de uma só vez. Caso escolha exibir os confrontos das oitavas de final, a função recolhe as informações da função *jogos_das_oitavas* e enumera os todos os confrontos de 1 a 8, conforme Imagem 12. Caso escolha exibir as médias de gols, é mostrado em tela a média total de gols da primeira fase, além das médias de gols de cada grupo, conforme Imagem 13. Caso escolha exibir a partida com maior goleada, a função recebe as informações da função *mais_gols* que determina qual partida houve a maior quantidade de gols por uma equipe e as informações da mesma, conforme Imagem 14.

```
#Função para exibir o relatório final
def relatorio(grupos, jogos):

    letras = list(grupos.keys())
    media_geral, medias = medias_jogos(jogos)
    partida = mais_gols(jogos)
    partidas = jogos_das_oitavas(grupos, jogos)

    escolha=menu('QUAL INFORMAÇÃO MOSTRAR','Confrontos das oitavas',

    if escolha==0:
        corda()
        print("OITAVAS DE FINAL\n")      #Partidas das oitavas
        for i in range(len(partidas)):
            print(f"{i+1} -> {partidas[i]}")
        print()
        corda()

        input('Pressione ENTER para continuar...')
        limpar_cmd()
```

Imagem 12. Função para exibir confronto das oitavas



```python
#Função para exibir o relatório final
def relatorio(grupos, jogos):

    letras = list(grupos.keys())
    media_geral, medias = medias_jogos(jogos)
    partida = mais_gols(jogos)
    partidas = jogos_das_oitavas(grupos, jogos)

    escolha=menu('QUAL INFORMAÇÃO MOSTRAR','Médias de gols','Maior goleada','Todas as

    if escolha==1:
        corda()
        print("ESTATÍSTICAS DA FASE DE GRUPO\n")     #Médias de gols
        print(f"Media de gols por jogo: {round(media_geral,2)}")
        for i in range(len(letras)):
            print(f"Média de gols do grupo {letras[i].upper()}: {round(medias[i],2)}")
        print()
        corda()
```

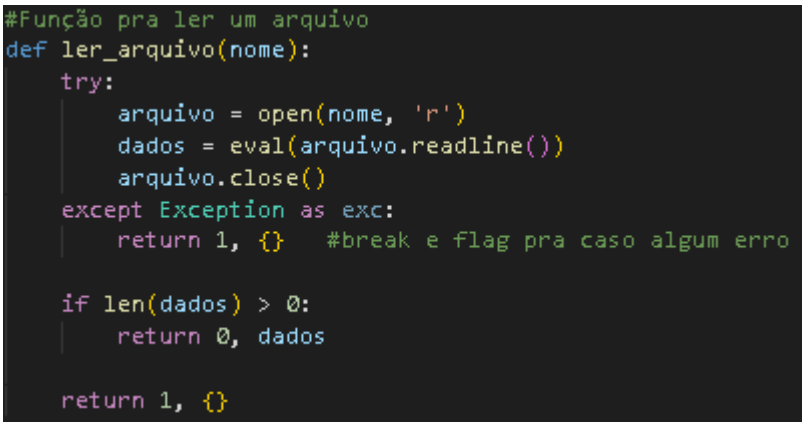
Imagem 13. Função para exibir média de gols da primeira fase

```python
#Função para exibir o relatório final
def relatorio(grupos, jogos):

    letras = list(grupos.keys())
    media_geral, medias = medias_jogos(jogos)
    partida = mais_gols(jogos)
    partidas = jogos_das_oitavas(grupos, jogos)

    escolha=menu('QUAL INFORMAÇÃO MOSTRAR','Médias de gols','Maior goleada','Todas as informações','Voltar')

    if escolha==2:
        corda()
        print("PARTIDA COM MAIOR GOLEADA\n")
        print(f"{partida['equipe 1'].title()} ({partida['placar'].split(' x ')[0]}) x ({partida['placar'].split(' x ')[1]})
        print(f"Ocorreu no estádio: {partida['estadio']} às {partida['horario']} do dia {partida['data']}")
        print(f"{partida['equipe 1'].title()} ficou com {partida['cartoes'].split(' x ')[0]} cartão(s) amarelo")
        print(f"{partida['equipe 2'].title()} ficou com {partida['cartoes'].split(' x ')[1]} cartão(s) amarelo\n")
        corda()

        input('Pressione ENTER para continuar...')
        limpar_cmd()
```

Imagem 14. Função para exibir partida com maior goleada

2.2.1.6. Funções para leitura e armazenamento das informações em arquivos

Primeiramente foi criada a função *ler_arquivo* para ter acesso às informações armazenadas em um arquivo por método texto. Ela possui uma verificação para caso ocorra algum erro na leitura do arquivo retornar “1”. Caso o tamanho da variável que recebe as informações do arquivo for maior que 0, ou seja, possua dado, a função retorna “0” e a própria variável, conforme Imagem 15.



```python
#Função pra ler um arquivo
def ler_arquivo(nome):
    try:
        arquivo = open(nome, 'r')
        dados = eval(arquivo.readline())
        arquivo.close()
    except Exception as exc:
        return 1, {}   #break e flag pra caso algum erro

    if len(dados) > 0:
        return 0, dados

    return 1, {}
```

Imagem 15. Função para leitura dos dados de um arquivo

Ademais, para armazenamento dos dados no arquivo por método texto, foi criada a função *escrever_arquivo* que possui a mesma verificação da função anterior (*ler_arquivo)*, e caso não possua nenhum erro, os dados armazenados em memória no formato de dicionário são passados como *string*, armazenados no arquivo desejado e retorna “0”, conforme Imagem 16.

```python
#Função para escrever em um arquivo
def escrever_arquivo(dado, nome):
    try:
        arquivo = open(nome, 'w')
        arquivo.write(str(dado))
        arquivo.close()
    except Exception:
        return 1, {}

    return 0
```

Imagem 16. Função para armazenamento dos dados de um arquivo

Para carregar os dados do arquivo escolhido para armazenar os dados do programa, foi criada a função *carregar_dados* que recebe o retorno da função *ler_arquivo* e caso não haja erro na leitura do arquivo em método texto, a função exibe em tela uma mensagem de confirmação de carregamento, transforma os dados armazenados em arquivo novamente em um dicionário e retorna, conforme Imagem 17.

```python
#Função para ler os dados do arquivo e tranformar em dicionário
def carregar_dados(grupos, jogos):
    erro, dado = ler_arquivo('grupos.txt')
    if erro == 0:
        print("Arquivo grupos.txt carregado com sucesso!")
        grupos = dict(dado)
    else:
        print("Não foi possível carregar o arquivo grupos.txt")

    erro, dado = ler_arquivo('jogos.txt')
    if erro == 0:
        print("Arquivo jogos.txt carregado com sucesso!")
        jogos = dict(dado)
    else:
        print("Não foi possível carregar o arquivo jogos.txt")

    return grupos, jogos
```

Imagem 17. Função para carregar os dados de um arquivo

Por fim, para o armazenamento dos dados em arquivo, foi desenvolvida a função *guardar_dados* que recebe o retorno da função *escrever_arquivo* e caso não haja nenhum erro na escrita dos dados no arquivo, a função exibe em tela uma mensagem de confirmação de armazenagem dos dados, conforme Imagem 18.

```python
#Guardar os dados dos jogos e dos grupos
def guardar_dados(grupos, jogos):
    erro = escrever_arquivo(grupos, 'grupos.txt')
    if erro == 0:
        print("Arquivo grupos.txt escrito com sucesso!")
    else:
        print("Não foi possível escrever o arquivo grupos.txt")

    erro = escrever_arquivo(jogos, 'jogos.txt')
    if erro == 0:
        print("Arquivo jogos.txt escrito com sucesso!\n")
    else:
        print("Não foi possível escrever o arquivo jogos.txt\n")
```

Imagem 18. Função para informar resultado da escrita dos dados em arquivo

2.2.1.7. Função principal

Para execução do programa e de todas as funções criadas e comentada anteriormente, foi criada a função *main*. Ela irá conter o dicionário dos grupos, definido com 8 chaves de A à H, onde cada chave irá receber uma lista com nome das quatro seleções disponíveis para cadastro em um grupo. Também irá conter um dicionário para os jogos, que recebe como chave a letra correspondente a cada grupo disponível. Dentro deste dicionário dos jogos, cada chave recebe outro dicionário que tem como chave os confrontos dos grupos, que tem como valor, as informações desses confrontos. Resumidamente, é um dicionário de dicionário de dicionário, conforme Imagem 19.

```
def main():
    grupos = {"A": [], "B": [], "C": [], "D": [], "E": [], "F": [], "G": [], "H": []}
    jogos = {}    #{letra do grupo: {partida: {dados}, partida2: {dados}, ...}}
    grupos, jogos = carregar_dados(grupos, jogos)
```

Imagem 19. Função principal com os dicionários dos grupos e jogos

Ademais, a função *main* conta com um loop para rodar o programa infinitamente até que seja solicitado seu encerramento. Ao ser solicitado, o programa grava as novas informações cadastradas ou editadas em arquivo e é encerrado. Segue demonstração na Imagem 20.

```
    escolha = -1
    while escolha != 4:
        escolha = menu('COPA DO MUNDO 2022\n', "Cadastrar", "Editar", "Visualizar", "Relatório Final" ,"Sair")
        if escolha == 4:
            print("Encerrando...\n")
        elif escolha == 0:
            cadastros(grupos, jogos)
            guardar_dados(grupos, jogos)
        elif escolha == 1:
            edicoes(grupos, jogos)
            guardar_dados(grupos, jogos)
        elif escolha == 2:
            visualizacoes(grupos, jogos)
        elif escolha == 3:
            if liberar_relatorio(jogos):
                relatorio(grupos, jogos)
            else:
                print("Os resultados ainda não foram lançados!")
                limpar_cmd(True)
    guardar_dados(grupos, jogos)
if __name__ == "__main__":
    main()
```

Imagem 20. Laço de repetição e condicionais para funcionamento do programa

3. Resultados e Discussões

O objetivo primário do programa é ser um software capaz de acompanhar as informações dos jogos da Copa do Mundo 2022 e ter acesso as suas estatísticas. Por isso, seu funcionamento foi desenvolvido para ser prático e objetivo, facilitando o seu manuseio e a experiência do usuário.

3.1. Manual de uso

Ao executar o programa, o usuário se depara com o menu inicial com algumas opções para escolha. Este menu permite escolher entre cadastrar algum jogo ou grupo, editar algum jogo ou grupo, visualizar os grupos ou jogos cadastrados, ter acesso ao relatório final com as estatísticas da primeira fase ou sair do programa, conforme Imagem 21.

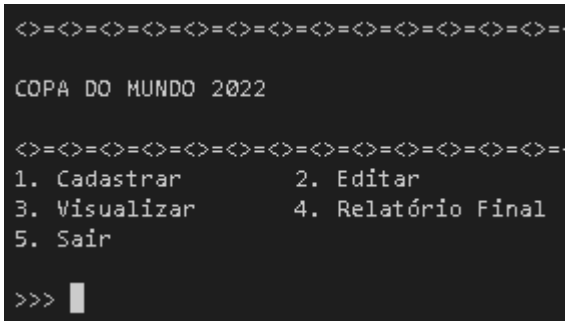


Imagem 21. Menu do programa

Ao escolher a opção “1”, as funcionalidades referentes ao cadastro dos dados são mostradas, conforme Imagem 22. É possível retornar ao menu escolhendo a opção “3”.

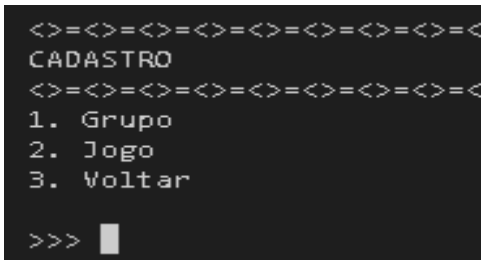


Imagem 22. Cadastro dos dados

Caso escolha cadastrar um grupo, é disponibilizado a escolha dos grupos de A à H, enumerados de 1 à 8. Caso escolha cadastrar um jogo, o usuário deve escolher também qual dos oito grupos cadastrar o jogo, onde lhe é mostrado em tela apenas os jogos que ainda não foram cadastrados. Em ambos os casos, se já terem sido cadastrados todos os grupos ou jogos, é informado ao usuário e o programa volta ao menu inicial.

Ao escolher a opção “2”, as funcionalidades referentes a edição dos dados são mostradas, conforme Imagem 23. É possível retornar ao menu escolhendo a opção “3”.

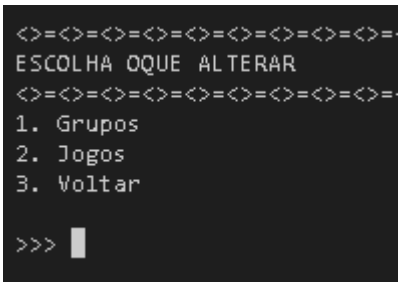


Imagem 23. Edição dos dados

Caso escolha editar um grupo, o usuário deve escolher um entre os grupos disponíveis e logo após, ele pode escolher entre limpar um grupo (excluir todas as seleções), trocar uma seleção (cadastrar uma nova seleção que ainda não tenha sido cadastrada em nenhum grupo) ou voltar ao menu, conforme Imagem 24.

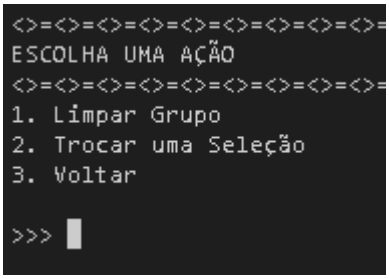


Imagem 24. Edição dos dados dos grupos

Caso escolha editar um grupo, o usuário deve escolher um entre os grupos disponíveis, um dos jogos cadastrados desse grupo e logo após ele pode escolher entre editar alguma informação desse jogo (estádio, placar, data, horário, cartões), apagar todas as informações desse jogo ou voltar ao menu, conforme Imagem 25.

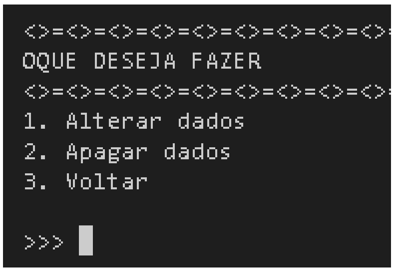


Imagem 25. Edição dos dados dos jogos

Ao escolher a opção “3”, as funcionalidades referentes a visualização dos dados cadastrados são exibidas, podendo escolher entre ter acesso aos grupos ou jogos cadastrados. É possível retornar ao menu escolhendo a opção “3”. Caso escolha visualizar os grupos cadastrados é exibido conforme exemplo na Imagem 26. Caso escolha visualizar os jogos cadastrados, é exibido conforme exemplo na Imagem 27.

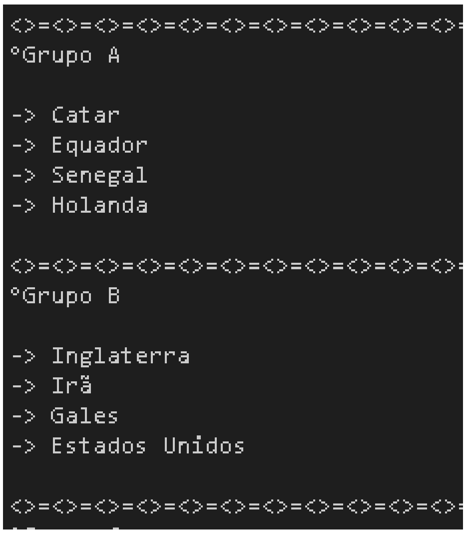


Imagem 26. Visualização dos grupos cadastrados

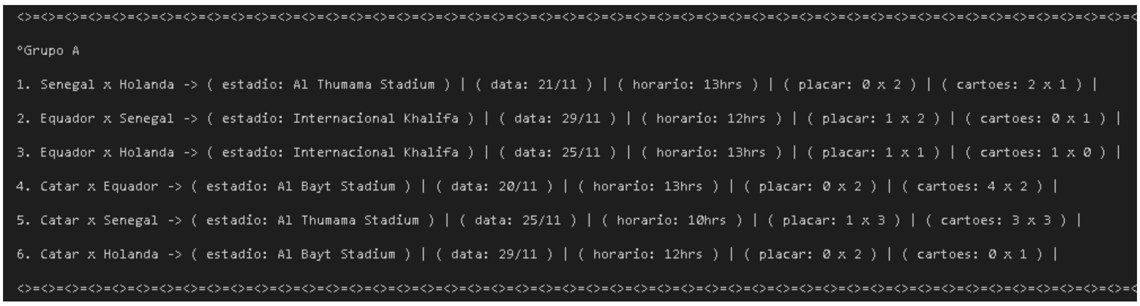


Imagem 27. Visualização dos jogos cadastrados por grupo

Ao escolher a opção “4”, as funcionalidades referentes a visualização das informações e estatísticas da primeira fase da Copa do Mundo 2022 são exibidas, conforme Imagem 28.

```
<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>
QUAL INFORMAÇÃO MOSTRAR
<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>
1. Confrontos das oitavas  2. Médias de gols
3. Maior goleada           4. Todas as informações
5. Voltar

>>> 
```

Imagem 28. Visualização do relatório final

O usuário tem a liberdade entre escolher visualizar cada informação individualmente, escolher a opção “4” e ter acesso a todas as informações disponíveis, conforme Imagem 29, ou voltar ao menu inicial.

```
<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<
OITAVAS DE FINAL

1 -> Holanda x Estados Unidos
2 -> Inglaterra x Senegal
3 -> Argentina x Austrália
4 -> França x Polônia
5 -> Japão x Croácia
6 -> Marrocos x Espanha
7 -> Brasil x Coreia Do Sul
8 -> Portugal x Suíça

<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<
ESTATÍSTICAS DA FASE DE GRUPO

Media de gols por jogo: 2.5
Média de gols do grupo A: 2.5
Média de gols do grupo B: 2.67
Média de gols do grupo C: 2.0
Média de gols do grupo D: 1.83
Média de gols do grupo E: 3.67
Média de gols do grupo F: 1.83
Média de gols do grupo G: 2.67
Média de gols do grupo H: 2.83

<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<
PARTIDA COM MAIOR GOLEADA

Espanha (7) x (0) Costa Rica
Ocorreu no estádio: Al Thumama Stadium às 13hrs do dia 23/11
Espanha ficou com 0 cartão(s) amarelo
Costa Rica ficou com 2 cartão(s) amarelo

<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<>=<
Pressione ENTER para continuar...
```

Imagem 29. Informações da primeira fase

Ao escolher a opção “5”, as informações são armazenadas em arquivos e o programa é encerrado por completo.

3.2. Testes e erros

Após a conclusão do código-fonte foram realizados testes visando a identificação de erros ou mau funcionamento do programa. Não foram constatados nenhum.

4. Conclusão

Com o desenvolvimento do software, foi possível colocar em prática vários conceitos da linguagem de programação Python, com destaque para: integridade de dados; manipulação de informações em arquivos; variáveis estruturadas heterogêneas; consistência de informações em registro; entre outros.

Os objetivos propostos para o desenvolvimento do software foram cumpridos, com a adição de algumas melhorias. Foi implementado uma opção que permite ao usuário ter acesso aos grupos e jogos já cadastrados. Esta funcionalidade possibilita que o usuário tenha controle de quais dados foram lançados e quais faltam lançar. É possível melhorar ainda mais o software do aplicativo de acompanhamento à Copa do Mundo 2022 otimizando ainda mais o código-fonte e desenvolvendo melhor o front-end dele, deixando-o visualmente mais interativo e chamativo. Vale também considerar futuramente adicionar o cadastro das informações das fases seguintes (quartas de final, semifinal e final), deixando assim o programa mais completo. Assim o aplicativo obteria captação total das informações deste importante evento.

5. Bibliografia Consultada

DANIEL NEVES SILVA (2022). “Copa do Mundo”. Disponível em: https://mundoeducacao.uol.com.br/educacao-fisica/copa-mundo.htm. Acesso em: 10 dez. 2022.

PAULA RONDINELLI (2022). “O fenômeno da Copa do Mundo”. Disponível em: https://brasilescola.uol.com.br/educacao-fisica/historia-da-copa-do-mundo.htm#:~:text=Import%C3%A2ncia%20da%20Copa%20do%20Mundo&text=Essa%20visibilidade%20mundial%20que%20a,maior%20oportunidade%20publicit%C3%A1ria%20do%20mundo. Acesso em: 10 dez. 2022..

SIGNIFICADOS (2017). “Copa do Mundo da FIFA”. Disponível em: https://www.significados.com.br/copa-do-mundo/. Acesso em: 10 dez. 2022.

GE GLOBO (2022). “Copa do Mundo da FIFA”. Disponível em: https://ge.globo.com/futebol/copa-do-mundo/2022/. Acesso em: 10 dez. 2022.

REDAÇÃO DO GE (2022). “Copa do Mundo 2022: datas e horários de todos os jogos”. Disponível em: https://ge.globo.com/futebol/copa-do-mundo/noticia/2022/04/01/horarios-jogos-copa-do-mundo-2022.ghtml. Acesso em: 10 dez. 2022.

GOOGLE (2022). “Copa do Mundo da FIFA Catar 2022”. Disponível em: https://www.google.com/search?q=jogos+copa+do+mundo&oq=jogos+copa+do+mundo&aqs=chrome..69i57j69i59j69i61.4706j0j7&sourceid=chrome&ie=UTF-8#sie=lg;/m/0fp_8fm;2;/m/030q7;mt;fp;1;;;. Acesso em: 10 dez. 2022.