instituto politécnico de gestão e tecnologia
escola superior de tecnologia

LICENCIATURA EM ENGENHARIA INFORMÁTICA

FUNDAMENTOS DE PROGRAMAÇÃO

LIGA DE FUTEBOL

FILIPE M. PINTO / JOÃO N. RODRIGUES

A22510400 / A22501188

GAIA
2026

Índice

Apresentação do sistema

O presente trabalho prático foi desenvolvido no âmbito da unidade curricular de Fundamentos de Programação e tem como objetivo a criação de um sistema de gestão de uma liga de futebol, em particular focamo-nos na *Premier League*.

A aplicação permite a gestão de jogos e equipas, recorrendo à persistência de dados através de ficheiros CSV e à importação automática de dados via API (football-data.org), cumprindo os requisitos de tratamento de dados reais.

As principais funcionalidades do sistema incluem:

- **Importação de Dados:** Ligação a uma API externa para carregar jogos reais.
- **Gestão de Jogos:** Inserção manual, edição e eliminação de registos de jogos.
- **Persistência:** Capacidade de guardar e carregar a base de dados em formato CSV (Comma Separated Values).
- **Estatísticas e Consultas:** Cálculo automático da tabela de classificação (pontos, vitórias, empates, derrotas), identificação da equipa com mais golos marcados e da equipa com a pior defesa.
- **Pesquisa:** Localização rápida de jogos específicos por jornada ou equipa.

O sistema foi desenvolvido em Python, utilizando uma arquitetura modular para facilitar a manutenção e leitura do código.

Fluxograma

Para a representação esquemática dos algoritmos desenvolvidos, optou-se pela geração de fluxogramas diretamente a partir da lógica do código implementado. Esta abordagem foi realizada com recurso à ferramenta **PlantUML**, uma linguagem de modelação aberta que permite criar diagramas através de descrições textuais.

A utilização desta metodologia assegura uma consistência total entre a solução codificada em Python e a sua representação gráfica, garantindo que os fluxogramas abaixo apresentados refletem fielmente o fluxo de execução, as estruturas de decisão e os ciclos utilizados no programa 'Liga de Futebol'."

Função: *main()*

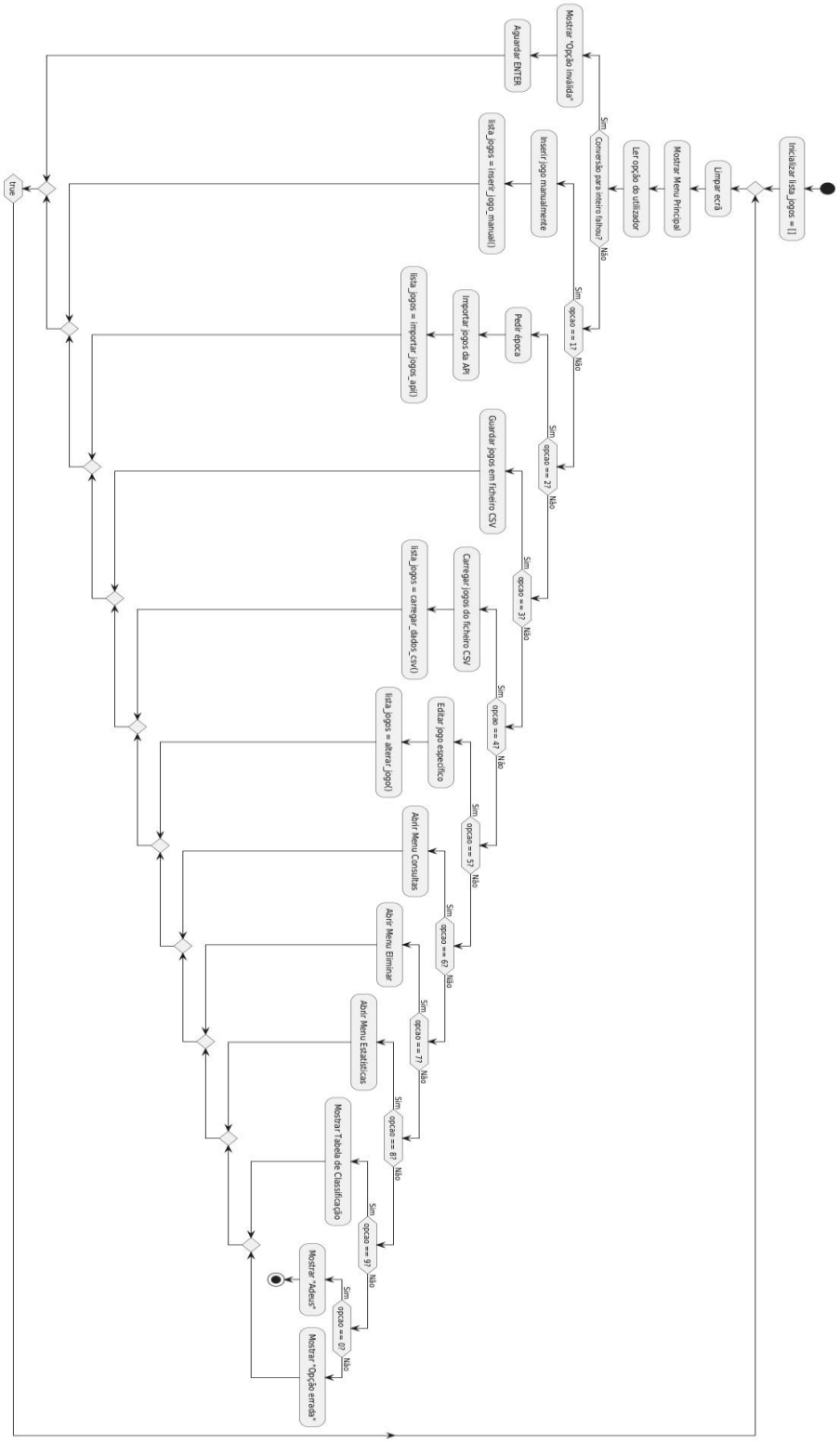


Figura 1 - fluxogramra main()

Função: *ler_data()*

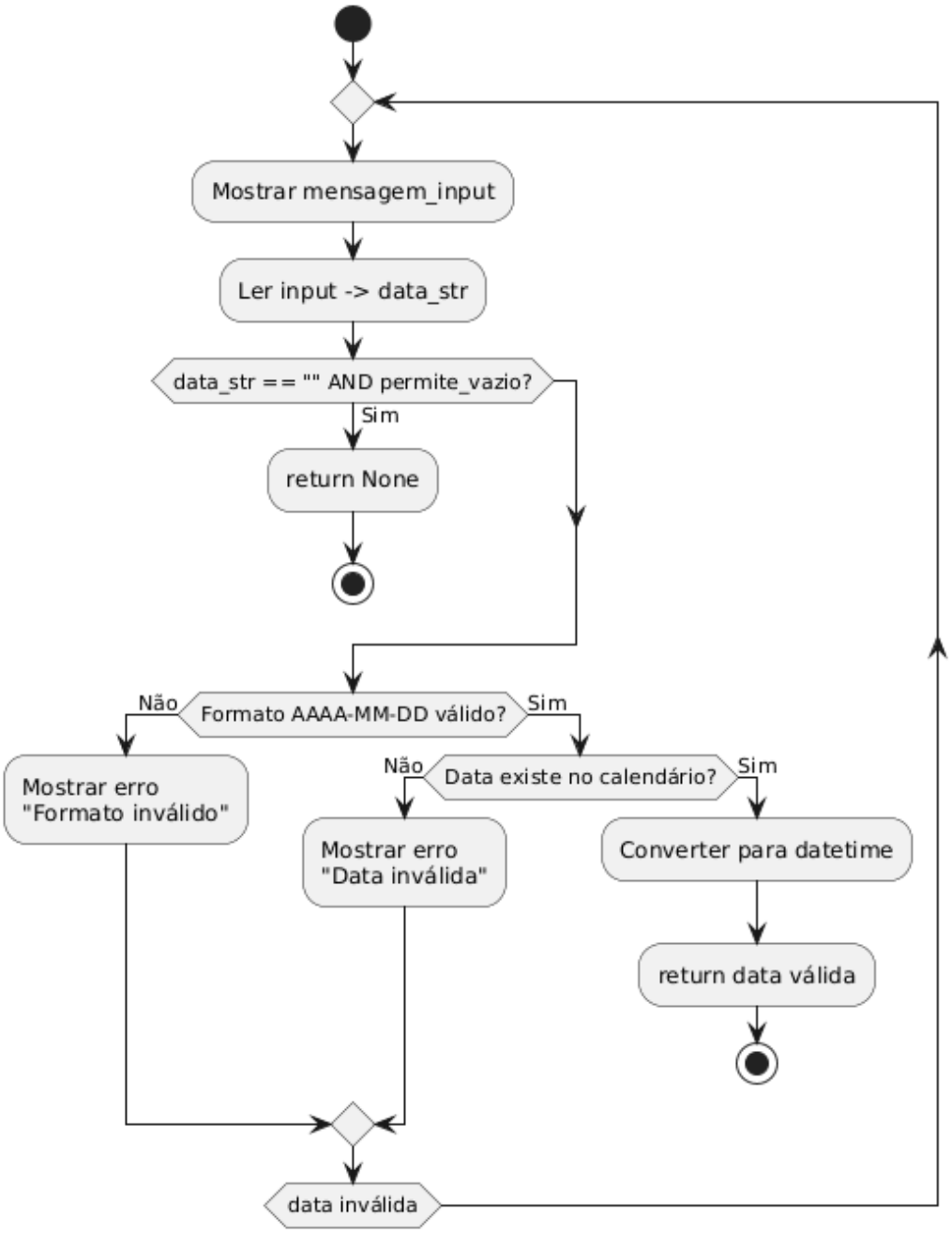


Figura 2 - Fluxograma ler_data()

Função: *ler_texto()*

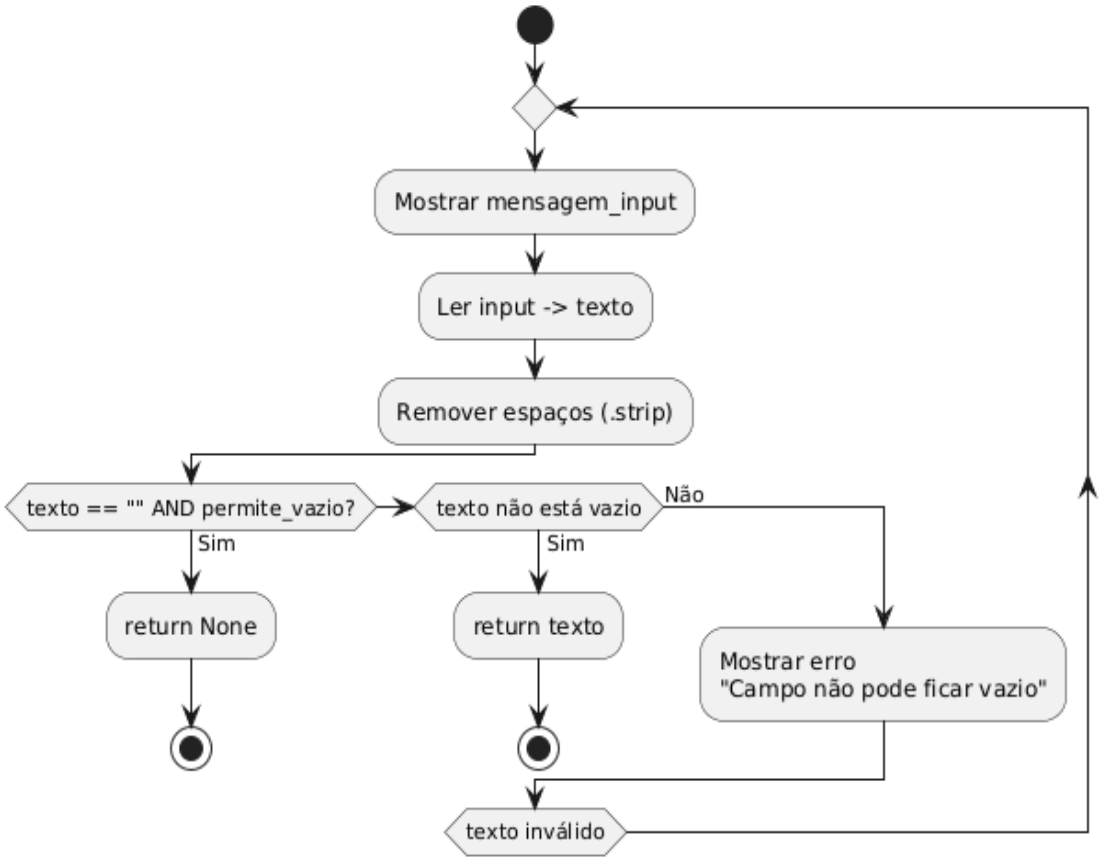


Figura 3 – fluxograma ler_texto()

Função: *ler_inteiro()*

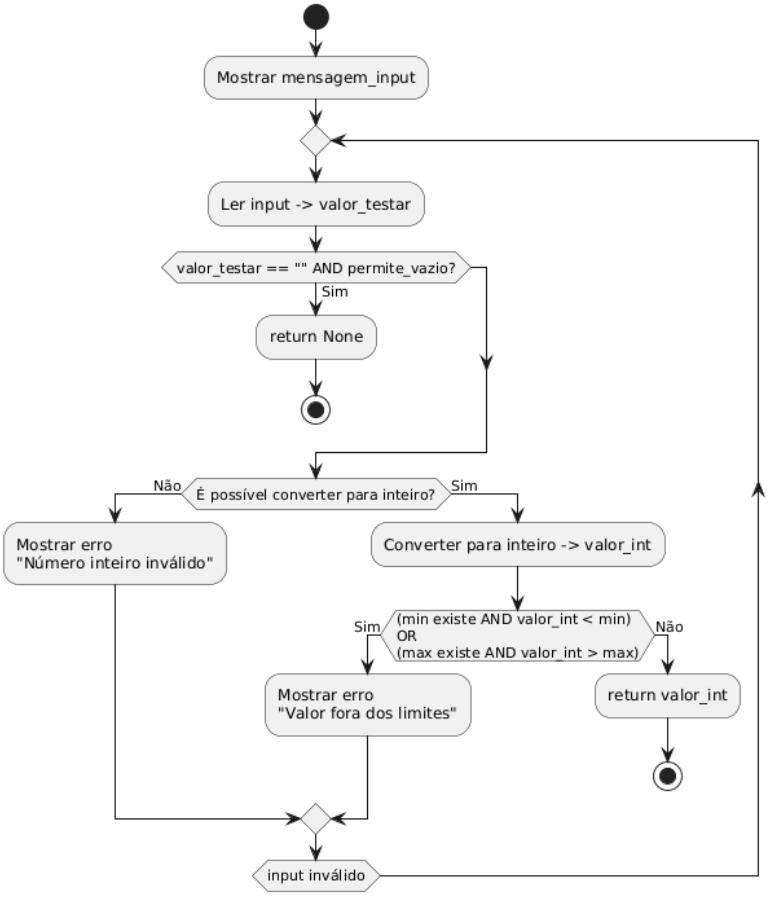


Figura 4 - fluxograma ler_interiro()

Função: *importar_jogos_api(api_key, época*)

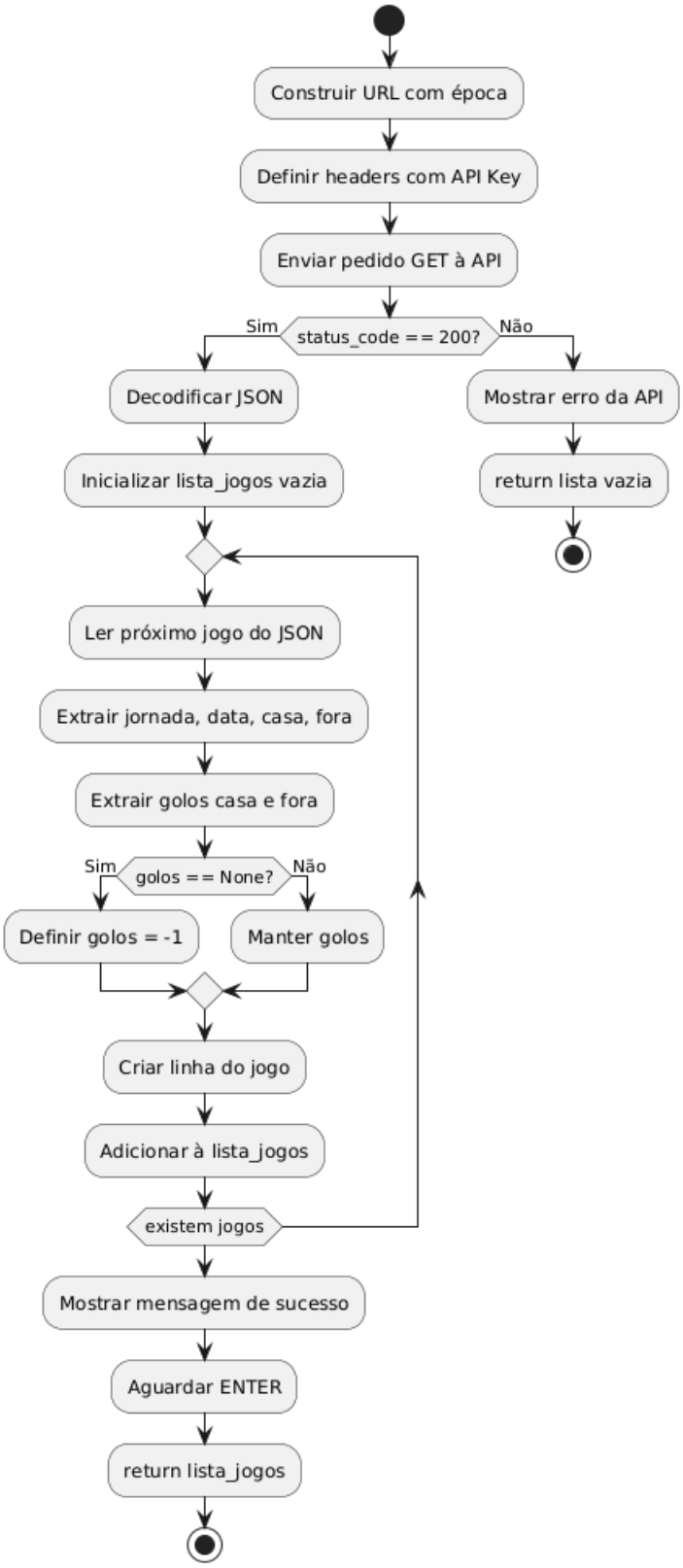


Figura 5 - fluxograma importar_jogos_api(api_key, época)

Função: *guardar_dados_csv(nome_ficheiro, lista_dados)*

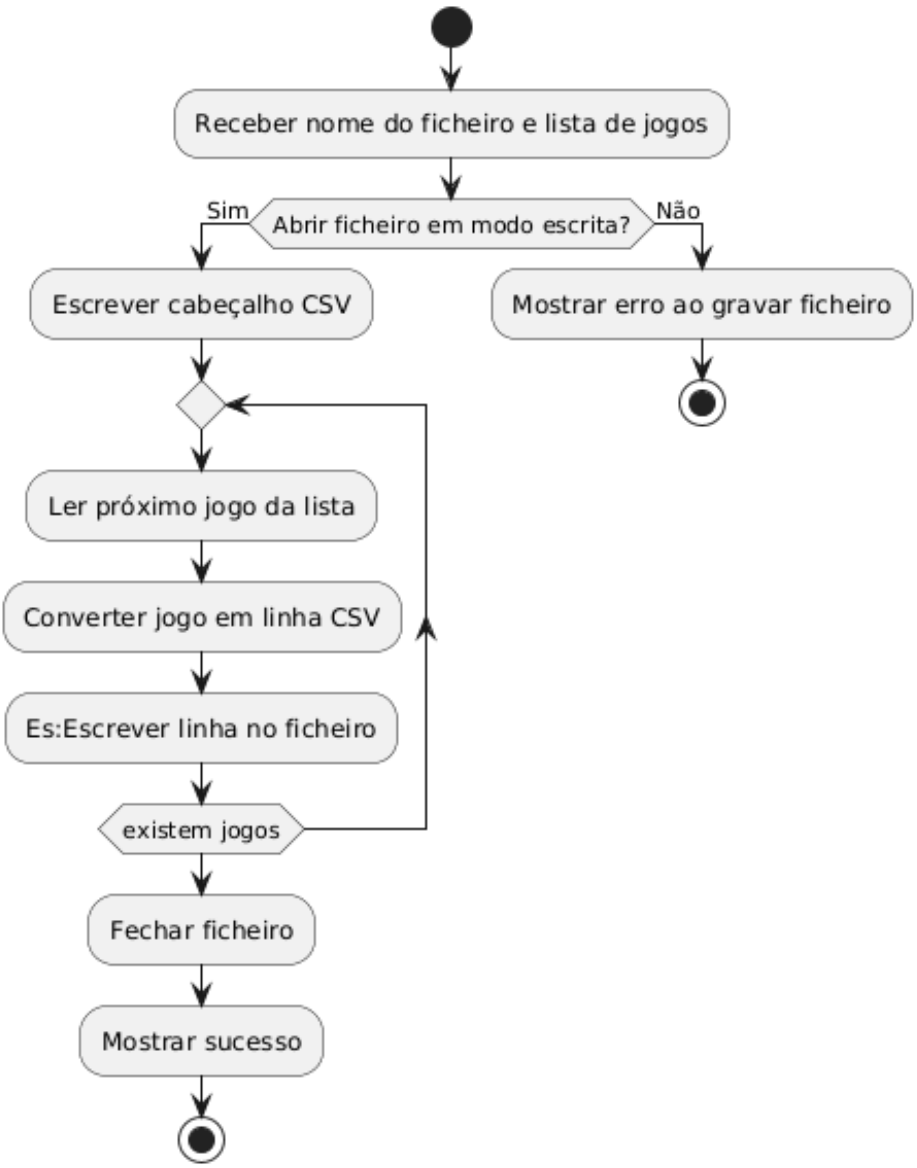


Figura 6 - fluxograma guardar_dados_csv(nome_ficheiro, lista_dados)

Função: *carregar_dados_csv(nome_ficheiro)*

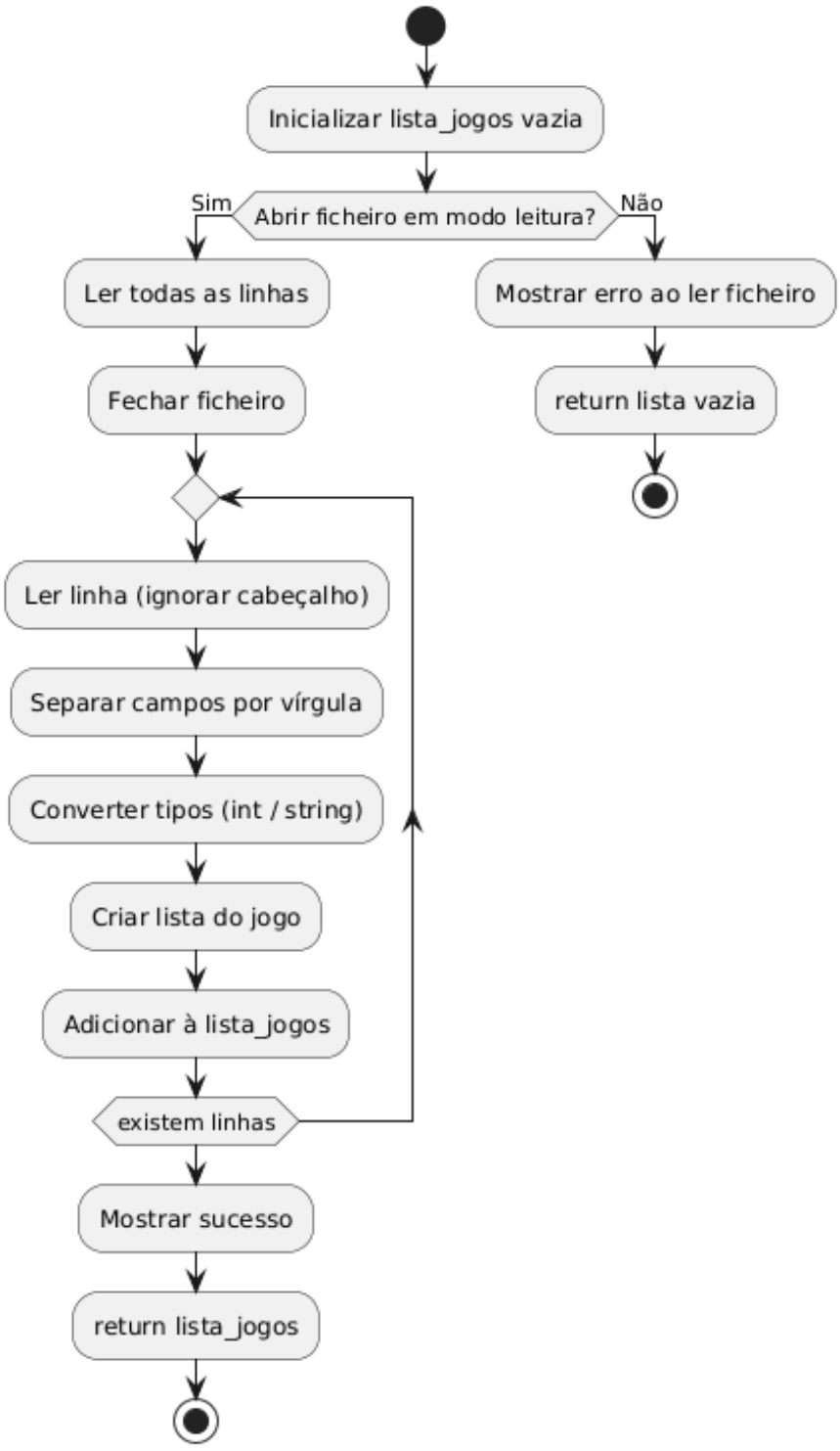


Figura 7 - fluxograma carregar_dados_csv(nome_ficheiro)

Função: *inserir_jogo_manual(lista_jogos)*

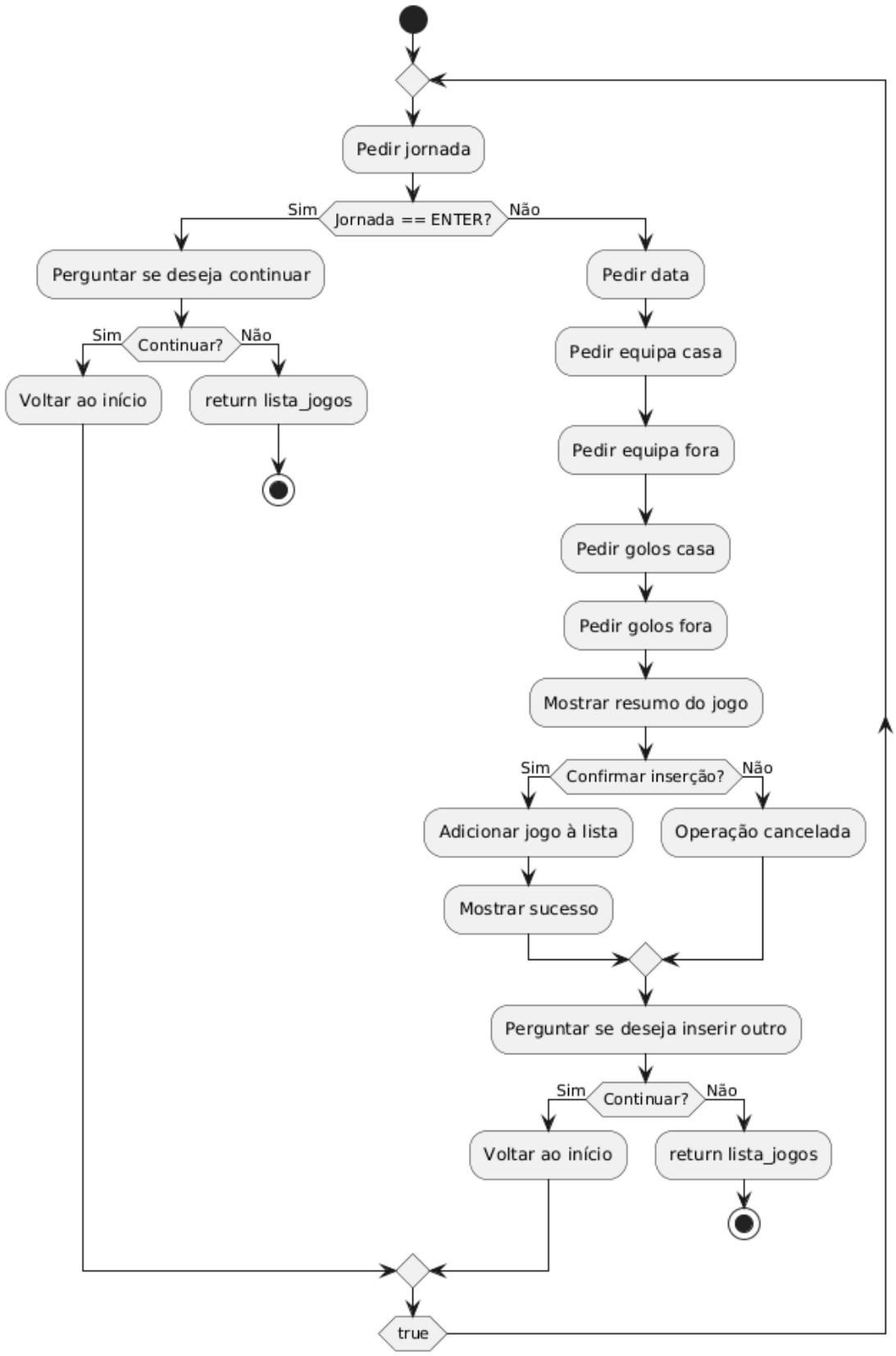


Figura 8 - inserir_jogo_manual(lista_jogos)

Função: eliminar_tudo()

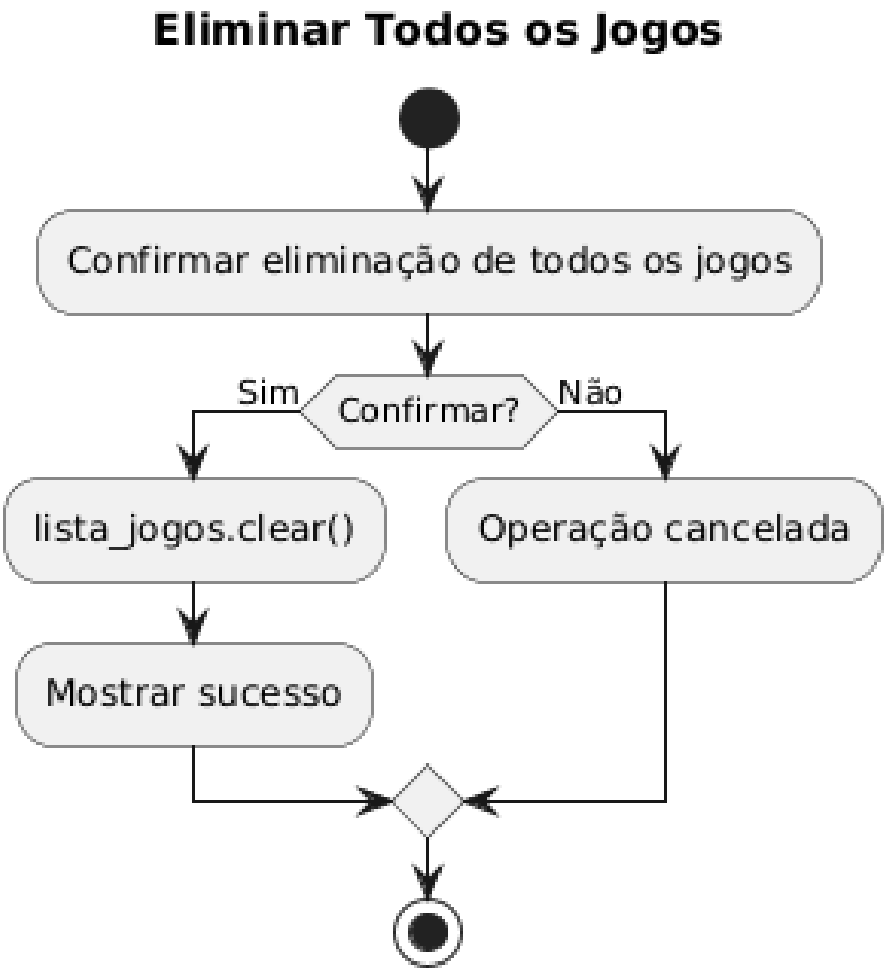


Figura 9 - fluxograma eliminar_tudo()

Função: *pesquisar_jogo(lista_jogos)*

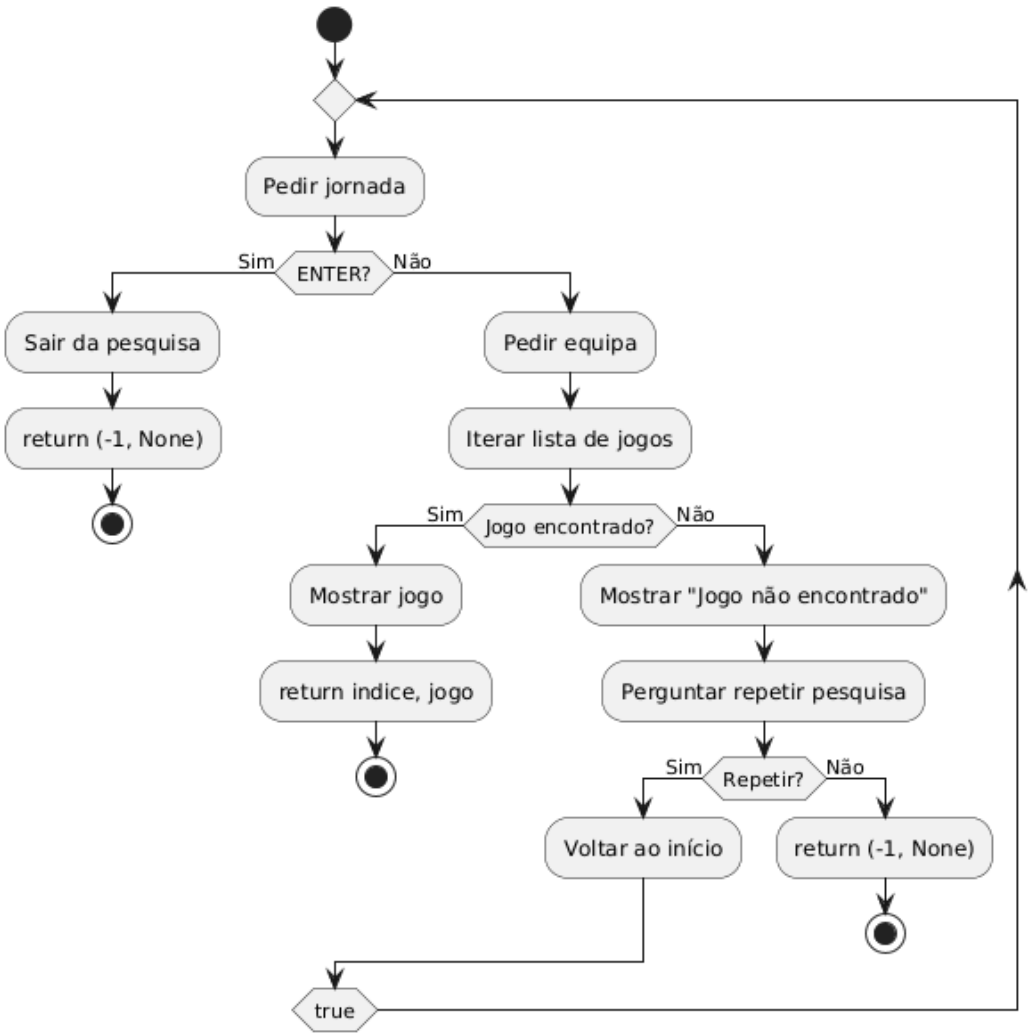


Figura 10 - fluxograma pesquisar_jogo(lista_jogos)

Função: alterar_jogo()

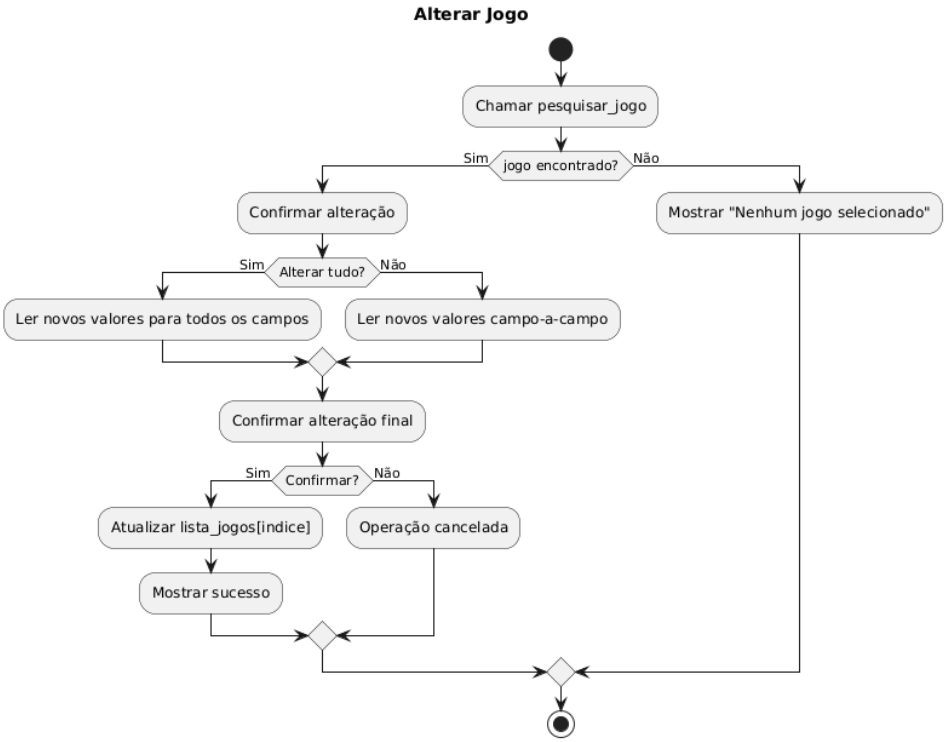


Figura 11 - fluxograma alterar_jogo()

Função: eliminar_jogo()

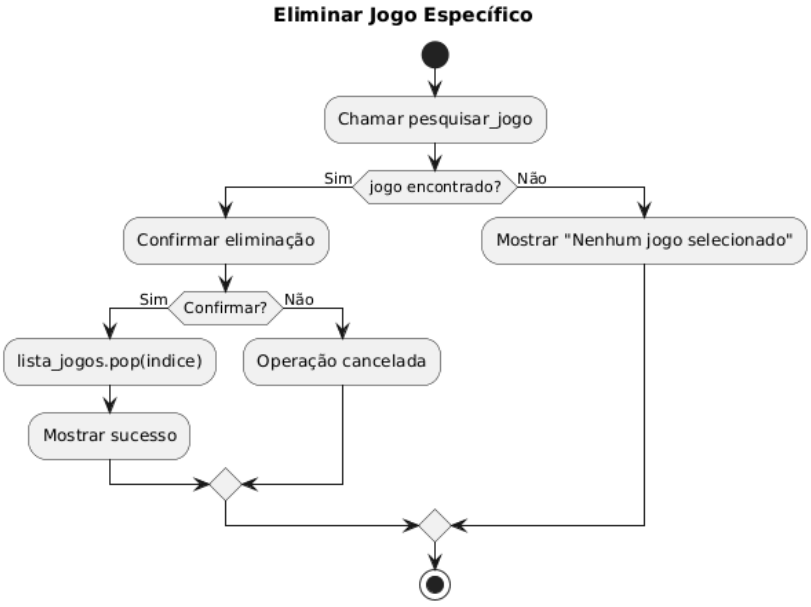


Figura 12 - fluxograma eliminar_jogo()

Função: consultar_todos_jogos():

Figura 13 - fluxograma consultar_todos_jogos()

Menu Estatística e funções de estatística:

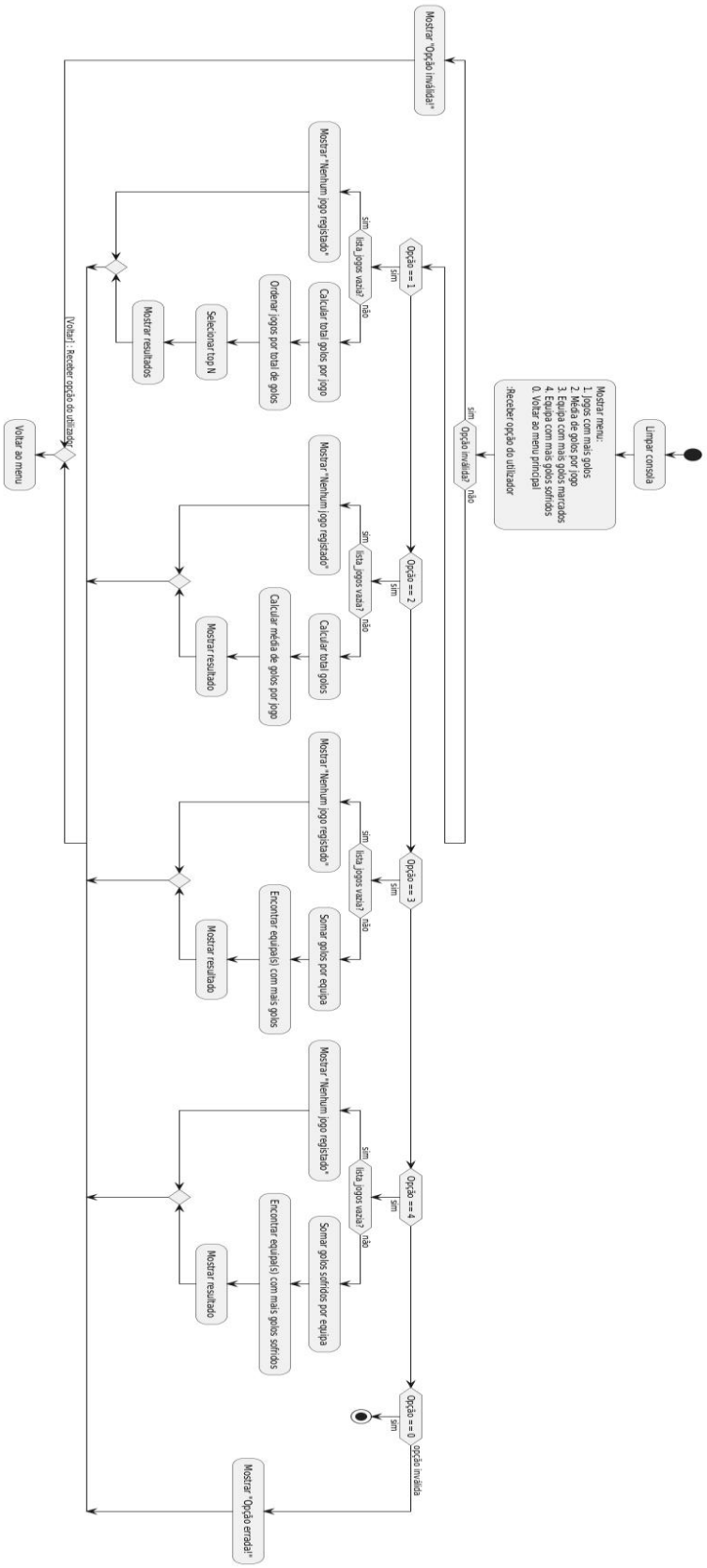


Figura 14 - -Fluxograma Menu e funções de estatística

Função: menu_consultar()

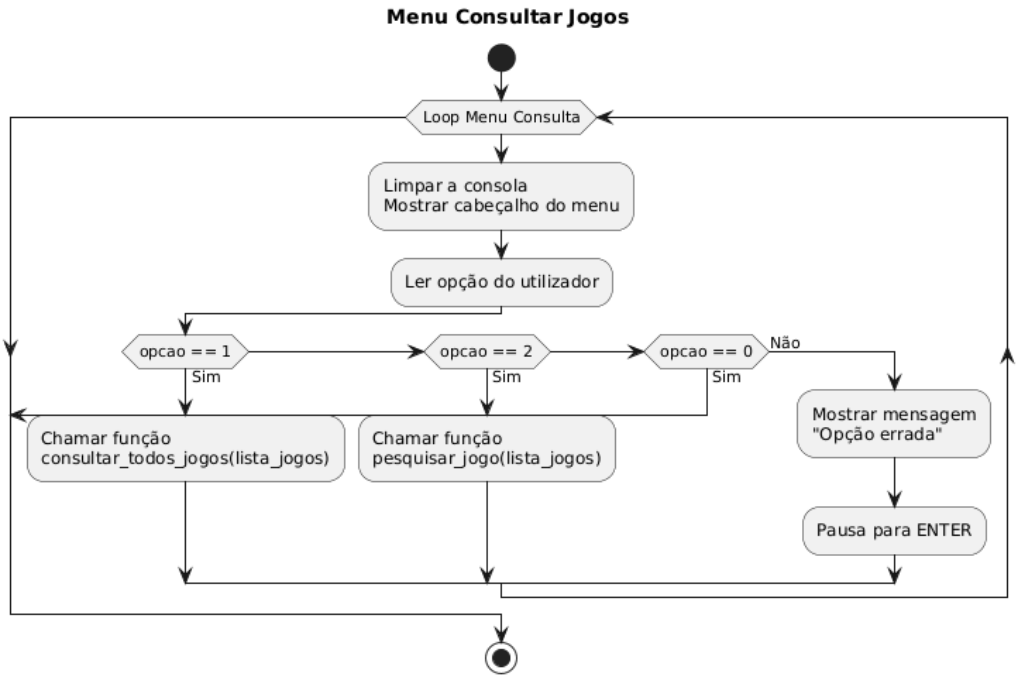


Figura 15 - fluxograma menu_consultar()

Função: menu_eliminar()

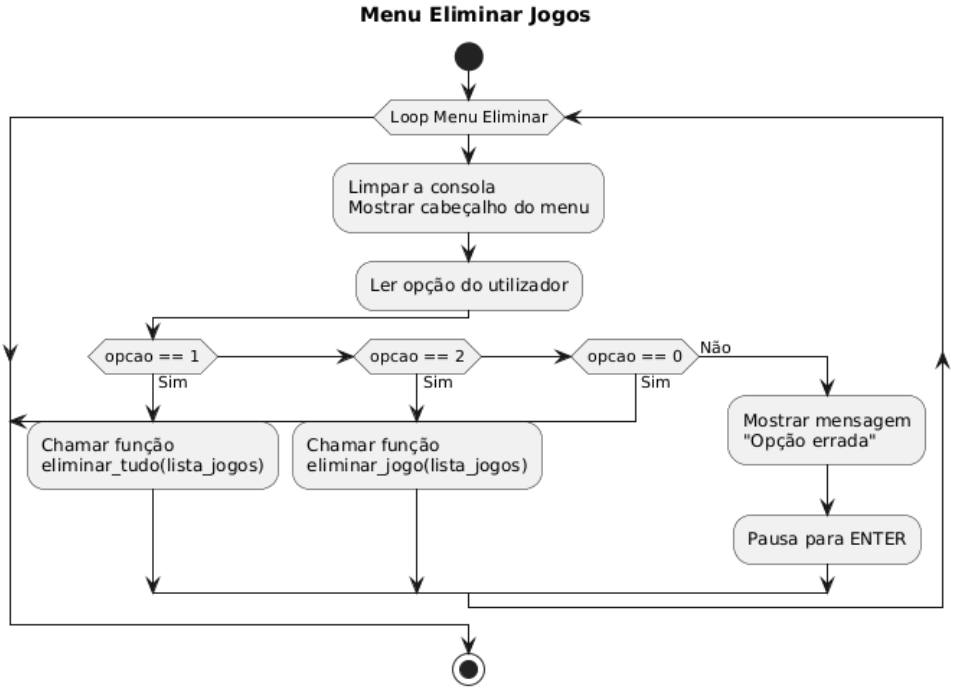


Figura 16 - fluxograma men_eliminar()

Pseudocódigo

Algoritmo: main()

```
ALGORITMO main

    IMPORTAR módulos:
        dados
        estatistica
        sistema operativo (os)

    DEFINIR chave_api

    lista_jogos ← LISTA VAZIA

    ENQUANTO VERDADEIRO FAZER

        LIMPAR ecrã
        MOSTRAR título "LIGA DE FUTEBOL – Menu Principal"

        MOSTRAR opções:
            1 – Introduzir jogo manualmente
            2 – Importar dados da API
            3 – Guardar dados em ficheiro
            4 – Carregar dados de ficheiro
            5 – Editar jogo específico
            6 – Menu de consultas
            7 – Menu de eliminar
            8 – Menu de estatísticas
            9 – Tabela de classificação
            0 – Sair

        TENTAR
            LER opção do utilizador

            SE opção = 1 ENTÃO
                lista_jogos ← inserir_jogo_manual(lista_jogos)

            SENÃO SE opção = 2 ENTÃO
                PEDIR época
                lista_jogos ← importar_jogos_api(chave_api, época)

            SENÃO SE opção = 3 ENTÃO
                guardar_dados_csv("jogos_premier_league.csv",
lista_jogos)

            SENÃO SE opção = 4 ENTÃO
```

```
                lista_jogos                                          ←
carregar_dados_csv("jogos_premier_league.csv")

            SENÃO SE opção = 5 ENTÃO
                lista_jogos ← alterar_jogo(lista_jogos)

            SENÃO SE opção = 6 ENTÃO
                menu_consultar(lista_jogos)

            SENÃO SE opção = 7 ENTÃO
                menu_eliminar(lista_jogos)

            SENÃO SE opção = 8 ENTÃO
                menu_estatisticas(lista_jogos)

            SENÃO SE opção = 9 ENTÃO
                tabela_classificacao(lista_jogos)

            SENÃO SE opção = 0 ENTÃO
                ESCREVER "Adeus"
                TERMINAR CICLO

            SENÃO
                ESCREVER "Opção errada"
            FIM SE

        EXCEPÇÃO
            ESCREVER "Opção inválida"
            AGUARDAR ENTER
        FIM TENTAR

    FIM ENQUANTO

FIM ALGORITMO
```

Algoritmo: importar_jogos_api(api_key, epoca)

```
ALGORITMO importar_jogos_api(api_key, epoca)

    CONSTRUIR url COM a época pretendida
    DEFINIR headers COM a chave da API

    ESCREVER "A ligar à API da Premier League..."
    resposta ← EXECUTAR pedido GET(url, headers)

    SE resposta.status_code = 200 ENTÃO
        dados_json ← DECODIFICAR resposta em JSON
        lista_jogos ← LISTA VAZIA

        PARA CADA jogo EM dados_json["matches"] FAZER
            jornada ← jogo.matchday
            data ← primeiros 10 caracteres de jogo.utcDate
            equipa_casa ← jogo.homeTeam.name
            equipa_fora ← jogo.awayTeam.name

            golos_casa ← jogo.score.fullTime.home
            golos_fora ← jogo.score.fullTime.away

            SE golos_casa É NULO ENTÃO
                golos_casa ← -1
                golos_fora ← -1
            FIM SE

            linha   ←   [jornada,   data,   equipa_casa,   golos_casa,
golos_fora, equipa_fora]
            ADICIONAR linha À lista_jogos
        FIM PARA

        ESCREVER "Importação concluída com sucesso"
        AGUARDAR ENTER
        DEVOLVER lista_jogos
    SENÃO
        ESCREVER "Erro na API"
        DEVOLVER LISTA VAZIA
    FIM SE

FIM ALGORITMO
```

Algoritmo: guardar_dados_csv(nome_ficheiro, lista_dados)

```
ALGORITMO guardar_dados_csv(nome_ficheiro, lista_dados)
    TENTAR
        ABRIR ficheiro EM modo escrita
        ESCREVER cabeçalho CSV

        PARA CADA jogo EM lista_dados FAZER
            linha ← CONCATENAR campos do jogo separados por vírgulas
            ESCREVER linha NO ficheiro
        FIM PARA

        FECHAR ficheiro
        ESCREVER "Ficheiro gravado com sucesso"
        AGUARDAR ENTER
    EXCEPÇÃO
        ESCREVER "Erro ao gravar ficheiro"
    FIM TENTAR

FIM ALGORITMO
```

Algoritmo: carregar_dados_csv(nome_ficheiro)

```
ALGORITMO carregar_dados_csv(nome_ficheiro)
    lista_jogos ← LISTA VAZIA
    TENTAR
        ABRIR ficheiro EM modo leitura
        linhas ← LER todas as linhas do ficheiro
        FECHAR ficheiro

        PARA i DE 1 ATÉ tamanho(linhas) - 1 FAZER
            linha ← REMOVER ENTER do fim da linha
            dados ← SEPARAR linha por vírgulas

            jornada ← CONVERTER dados[0] PARA inteiro
            data ← dados[1]
            equipa_casa ← dados[2]
            golos_casa ← CONVERTER dados[3] PARA inteiro
            golos_fora ← CONVERTER dados[4] PARA inteiro
            equipa_fora ← dados[5]

            jogo ← [jornada, data, equipa_casa, golos_casa, golos_fora,
equipa_fora]
            ADICIONAR jogo À lista_jogos
        FIM PARA

        ESCREVER "Ficheiro carregado com sucesso"
        DEVOLVER lista_jogos
    EXCEPÇÃO
        ESCREVER "Erro ao ler ficheiro"
        DEVOLVER LISTA VAZIA
    FIM TENTAR

FIM ALGORITMO
```

Algoritmo: inserir_jogo_manual(lista_jogos)

```
ALGORITMO inserir_jogo_manual(lista_jogos)

    ENQUANTO VERDADEIRO FAZER

        PEDIR jornada (permite ENTER)

        SE jornada É NULO ENTÃO
            PERGUNTAR "Deseja inserir outro jogo?"
            SE resposta ≠ "S" ENTÃO
                DEVOLVER lista_jogos
            SENÃO
                CONTINUAR
            FIM SE
        FIM SE

        PEDIR data
        PEDIR equipa_casa
        PEDIR equipa_fora
        PEDIR golos_casa
        PEDIR golos_fora

        novo_jogo ← [jornada, data, equipa_casa, golos_casa,
golos_fora, equipa_fora]

        MOSTRAR resumo do jogo
        PERGUNTAR confirmação

        SE confirmação ≠ "S" ENTÃO
            PERGUNTAR se deseja continuar
            SE resposta ≠ "S" ENTÃO
                DEVOLVER lista_jogos
            SENÃO
                CONTINUAR
            FIM SE
        FIM SE

        ADICIONAR novo_jogo À lista_jogos
        MOSTRAR lista de jogos

        PERGUNTAR se deseja inserir outro jogo
        SE resposta ≠ "S" ENTÃO
            DEVOLVER lista_jogos
        FIM SE

    FIM ENQUANTO

FIM ALGORITMO
```

Algoritmo: pesquisar_jogo(lista_jogos)

```
ALGORITMO pesquisar_jogo(lista_jogos)
    ENQUANTO VERDADEIRO FAZER
        PEDIR jornada (permite ENTER)

        SE jornada É NULO ENTÃO
            DEVOLVER (-1, NULO)
        FIM SE

        PEDIR equipa

        PARA i DE 0 ATÉ tamanho(lista_jogos) - 1 FAZER
            jogo ← lista_jogos[i]

            SE jornada = jogo.jornada E
               (equipa PERTENCE a jogo.casa OU jogo.fora) ENTÃO
                MOSTRAR jogo encontrado
                AGUARDAR ENTER
                DEVOLVER (i, jogo)
            FIM SE
        FIM PARA

        MOSTRAR "Jogo não encontrado"
        PERGUNTAR repetir pesquisa

        SE resposta ≠ "S" ENTÃO
            DEVOLVER (-1, NULO)
        FIM SE

    FIM ENQUANTO

FIM ALGORITMO
```

Algoritmo: alterar_jogo(lista_jogos)

```
ALGORITMO alterar_jogo(lista_jogos)

    (indice, jogo_original) ← pesquisar_jogo(lista_jogos)

    SE indice = -1 ENTÃO
        ESCREVER "Nenhum jogo selecionado"
        DEVOLVER lista_jogos
    FIM SE

    PERGUNTAR confirmação

    SE resposta ≠ "S" ENTÃO
        DEVOLVER lista_jogos
    FIM SE

    PERGUNTAR se deseja alterar todos os campos

    SE resposta = "S" ENTÃO
        PEDIR todos os campos
        jogo_novo ← NOVO jogo
    SENÃO
        jogo_novo ← CÓPIA de jogo_original
        PARA CADA campo FAZER
            PEDIR novo valor (ENTER mantém)
            SE valor ≠ NULO ENTÃO
                ATUALIZAR campo
            FIM SE
        FIM PARA
    FIM SE

    PERGUNTAR confirmação final

    SE resposta = "S" ENTÃO
        SUBSTITUIR jogo na lista
        ESCREVER "Jogo alterado com sucesso"
    SENÃO
        ESCREVER "Alteração cancelada"
    FIM SE

    DEVOLVER lista_jogos

FIM ALGORITMO
```

Algoritmo: eliminar_jogo(lista_jogos)

```
ALGORITMO eliminar_jogo(lista_jogos)
    (indice, jogo) ← pesquisar_jogo(lista_jogos)

    SE indice = -1 ENTÃO
        DEVOLVER lista_jogos
    FIM SE

    PERGUNTAR confirmação

    SE resposta = "S" ENTÃO
        REMOVER jogo da lista
        ESCREVER "Jogo apagado com sucesso"
        AGUARDAR ENTER
    SENÃO
        ESCREVER "Operação cancelada"
    FIM SE

    DEVOLVER lista_jogos

FIM ALGORITMO
```

Algoritmo: eliminar_jogo(lista_jogos)

```
ALGORITMO eliminar_jogo(lista_jogos)

    (indice_jogo, descricao_jogo) ← pesquisar_jogo(lista_jogos)

    SE indice_jogo = -1 ENTÃO
        ESCREVER "Nenhum jogo selecionado"
        DEVOLVER lista_jogos
    FIM SE

    MOSTRAR descricao_jogo
    PERGUNTAR "Tem a certeza que quer apagar este jogo? (S/N)"

    SE resposta = "S" ENTÃO
        REMOVER jogo da lista_jogos NA posição indice_jogo
        ESCREVER "Jogo apagado com sucesso"
        AGUARDAR ENTER
    SENÃO
        ESCREVER "Operação cancelada"
    FIM SE

    DEVOLVER lista_jogos

FIM ALGORITMO
```

Algoritmo: consultar_todos_jogos(lista_jogos)

```
ALGORITMO consultar_todos_jogos(lista_jogos)

    LIMPAR ecrã
    ESCREVER "Consultar Todos os Jogos"
    ESCREVER nota sobre golos = -1
    AGUARDAR ENTER

    SE tamanho(lista_jogos) = 0 ENTÃO
        ESCREVER "Nenhum jogo registado"
        AGUARDAR ENTER
        TERMINAR
    FIM SE

    jogos_por_jornada ← DICIONÁRIO vazio

    PARA CADA jogo EM lista_jogos FAZER
        jornada ← jogo.jornada
        SE jornada NÃO EXISTE EM jogos_por_jornada ENTÃO
            jogos_por_jornada[jornada] ← LISTA vazia
        FIM SE
        ADICIONAR jogo A jogos_por_jornada[jornada]
    FIM PARA

    PARA CADA jornada EM jogos_por_jornada FAZER
        ESCREVER "Jornada", jornada
        PARA CADA jogo EM jogos_por_jornada[jornada] FAZER
            MOSTRAR data, equipa casa, equipa fora e resultado
        FIM PARA
    FIM PARA

    AGUARDAR ENTER

FIM ALGORITMO
```

Algoritmo: menu_consultar(lista_jogos)

```
ALGORITMO menu_consultar(lista_jogos)

    ENQUANTO VERDADEIRO FAZER
        LIMPAR ecrã
        MOSTRAR menu de consulta
        MOSTRAR opções:
            1 – Consultar todos os jogos
            2 – Consultar jogo específico
            0 – Voltar ao menu principal

        TENTAR
            LER opção
            SE opção = 1 ENTÃO
                consultar_todos_jogos(lista_jogos)
            SENÃO SE opção = 2 ENTÃO
                pesquisar_jogo(lista_jogos)
            SENÃO SE opção = 0 ENTÃO
                TERMINAR CICLO
            SENÃO
                ESCREVER "Opção errada"
            FIM SE
        EXCEPÇÃO
            ESCREVER "Opção inválida"
            AGUARDAR ENTER
        FIM TENTAR
    FIM ENQUANTO

FIM ALGORITMO
```

Algoritmo: menu_eliminar(lista_jogos)

```
ALGORITMO menu_eliminar(lista_jogos)

    ENQUANTO VERDADEIRO FAZER
        LIMPAR ecrã
        MOSTRAR menu de eliminação
        MOSTRAR opções:
            1 – Eliminar todos os jogos
            2 – Eliminar jogo específico
            0 – Voltar ao menu principal

        TENTAR
            LER opção
            SE opção = 1 ENTÃO
                eliminar_tudo(lista_jogos)
            SENÃO SE opção = 2 ENTÃO
                eliminar_jogo(lista_jogos)
            SENÃO SE opção = 0 ENTÃO
                TERMINAR CICLO
            SENÃO
                ESCREVER "Opção errada"
            FIM SE
        EXCEPÇÃO
            ESCREVER "Opção inválida"
            AGUARDAR ENTER
        FIM TENTAR
    FIM ENQUANTO

FIM ALGORITMO
```

Pseudocódigo – Validações e Estatísticas

Algoritmo: ler_inteiro(mensagem, min, max, permite_vazio)

```
ALGORITMO ler_inteiro(mensagem, min, max, permite_vazio)

    ENQUANTO VERDADEIRO FAZER
        PEDIR valor ao utilizador

        SE valor é vazio E permite_vazio = VERDADEIRO ENTÃO
            DEVOLVER NULO
        FIM SE

        TENTAR
            converter valor para inteiro

            SE (min existe E valor < min) OU (max existe E valor > max)
ENTÃO
                MOSTRAR mensagem de erro de intervalo
            SENÃO
                DEVOLVER valor inteiro
            FIM SE

        EXCEPÇÃO
            MOSTRAR "Erro: número inteiro inválido"
        FIM TENTAR
    FIM ENQUANTO

FIM ALGORITMO
```

Algoritmo: ler_texto(mensagem, permite_vazio)

```
ALGORITMO ler_texto(mensagem, permite_vazio)

    ENQUANTO VERDADEIRO FAZER
        PEDIR texto ao utilizador
        REMOVER espaços extras

        SE texto é vazio E permite_vazio = VERDADEIRO ENTÃO
            DEVOLVER NULO
        SENÃO SE texto não é vazio ENTÃO
            DEVOLVER texto
        SENÃO
            MOSTRAR "Erro: campo obrigatório"
        FIM SE
    FIM ENQUANTO

FIM ALGORITMO
```

Algoritmo: ler_data(mensagem, permite_vazio)

```
ALGORITMO ler_data(mensagem, permite_vazio)

    ENQUANTO VERDADEIRO FAZER
        PEDIR data ao utilizador

        SE data é vazia E permite_vazio = VERDADEIRO ENTÃO
            DEVOLVER NULO
        FIM SE

        TENTAR
            VALIDAR data no formato AAAA-MM-DD
            DEVOLVER data
        EXCEPÇÃO
            MOSTRAR "Erro: data inválida"
        FIM TENTAR
    FIM ENQUANTO

FIM ALGORITMO
```

Algoritmo: menu_estatisticas(lista_jogos)

```
ALGORITMO menu_estatisticas(lista_jogos)

    ENQUANTO VERDADEIRO FAZER
        LIMPAR ecrã
        MOSTRAR menu de estatísticas
        TENTAR
            LER opção

            SE opção = 1 ENTÃO
                jogos_mais_golos(lista_jogos)
            SENÃO SE opção = 2 ENTÃO
                media_golos_por_jogo(lista_jogos)
            SENÃO SE opção = 3 ENTÃO
                equipa_mais_golos(lista_jogos)
            SENÃO SE opção = 4 ENTÃO
                equipa_mais_golos_sofridos(lista_jogos)
            SENÃO SE opção = 0 ENTÃO
                SAIR DO CICLO
            SENÃO
                MOSTRAR "Opção errada"
            FIM SE
        EXCEPÇÃO
            MOSTRAR "Opção inválida"
            AGUARDAR ENTER
        FIM TENTAR
    FIM ENQUANTO

FIM ALGORITMO
```

Algoritmo: calcular_total_golos(jogo)

```
ALGORITMO calcular_total_golos(jogo)
    DEVOLVER golos_casa + golos_fora
FIM ALGORITMO
```

Algoritmo: jogos_mais_golos(lista_jogos)

```
ALGORITMO jogos_mais_golos(lista_jogos)
    SE lista_jogos está vazia ENTÃO
        MOSTRAR "Nenhum jogo registado"
        AGUARDAR ENTER
        DEVOLVER lista vazia
    FIM SE

    PARA CADA jogo EM lista_jogos FAZER
        calcular total de golos
        adicionar jogo com total à nova lista
    FIM PARA

    ORDENAR lista por total de golos (descendente)
    SELECIONAR os primeiros 10 jogos

    MOSTRAR jogos
    AGUARDAR ENTER

FIM ALGORITMO
```

Algoritmo: equipa_mais_golos(lista_jogos)

```
ALGORITMO equipa_mais_golos(lista_jogos)
    SE lista_jogos vazia ENTÃO
        MOSTRAR erro
        AGUARDAR ENTER
        TERMINAR
    FIM SE

    PARA CADA jogo FAZER
        somar golos marcados por cada equipa num dicionário
    FIM PARA

    DETERMINAR máximo de golos
    IDENTIFICAR equipa(s) com esse máximo

    MOSTRAR resultado
    AGUARDAR ENTER

FIM ALGORITMO
```

Algoritmo: equipa_mais_golos_sofridos(lista_jogos)

```
ALGORITMO equipa_mais_golos_sofridos(lista_jogos)

    SE lista_jogos vazia ENTÃO
        MOSTRAR erro
        AGUARDAR ENTER
        TERMINAR
    FIM SE

    PARA CADA jogo FAZER
        somar golos sofridos por cada equipa
    FIM PARA

    IDENTIFICAR equipa(s) com mais golos sofridos

    MOSTRAR resultado
    AGUARDAR ENTER

FIM ALGORITMO
```

Algoritmo: media_golos_por_jogo(lista_jogos)

```
ALGORITMO media_golos_por_jogo(lista_jogos)

    SE lista_jogos vazia ENTÃO
        MOSTRAR erro
        AGUARDAR ENTER
        DEVOLVER -1
    FIM SE

    PARA CADA jogo FAZER
        somar total de golos
    FIM PARA

    CALCULAR média
    MOSTRAR média
    AGUARDAR ENTER

FIM ALGORITMO
```

Algoritmo: tabela_classificacao(lista_jogos)

```
ALGORITMO tabela_classificacao(lista_jogos)

    SE lista_jogos vazia ENTÃO
        MOSTRAR erro
        AGUARDAR ENTER
        TERMINAR
    FIM SE

    PARA CADA jogo FAZER
        atualizar pontos e jogos de cada equipa
    FIM PARA

    CONVERTER dicionário para lista
    ORDENAR lista por pontos (descendente)

    MOSTRAR tabela de classificação
    AGUARDAR ENTER

FIM ALGORITMO
```

Prova e teste

Para validar o correto funcionamento do sistema "Liga de Futebol", foram realizados testes exaustivos cobrindo as quatro áreas principais da aplicação: Gestão de Dados, Persistência, Manipulação e Estatística.

Abaixo apresentam-se os cenários de teste desenhados, os dados de entrada utilizados e os resultados obtidos.

1. Teste de Inserção e Validação de Dados

Objetivo: Verificar se o sistema aceita dados válidos e se rejeita dados inválidos (proteção contra erros).

- **Cenário A: Inserção Manual de Jogo**
    - **Ação:** Selecionar opção 1 e inserir: Jornada 5, Data 2025-02-20, Casa Porto, Fora Benfica, Golos Casa 2, Golos Fora 1.
    - **Resultado Esperado:** O sistema deve confirmar a inserção e adicionar o jogo à lista.
    - **Resultado Obtido:** Sucesso. O jogo ficou registado na memória.

- **Cenário B: Validação de Erros (Inputs Inválidos)**
    - **Ação:** Tentar inserir a Jornada 40 (limite é 38) ou inserir texto "Trinta" no campo de golos.
    - **Resultado Esperado:** O módulo validacoes.py deve detetar o erro e pedir o valor novamente sem crashar o programa.

- **Resultado Obtido:** O sistema apresentou a mensagem "Erro: O valor tem de estar entre 1 e 38" para a jornada e "Erro: Tens de introduzir um número inteiro válido" para os golos, impedindo a gravação de dados corrompidos.

2. Teste de Importação via API (Dados Reais)

Objetivo: Validar a conectividade com o serviço externo *football-data.org*.

- **Cenário:**
  - **Ação:** Selecionar opção 2 (Importar API) e inserir a época 2024.

.

  - **Dados de Entrada:** Chave de API válida configurada no main.py.
  - **Resultado Esperado:** Código de estado HTTP 200, descodificação do JSON e preenchimento da lista.
    - **Exclusão:**

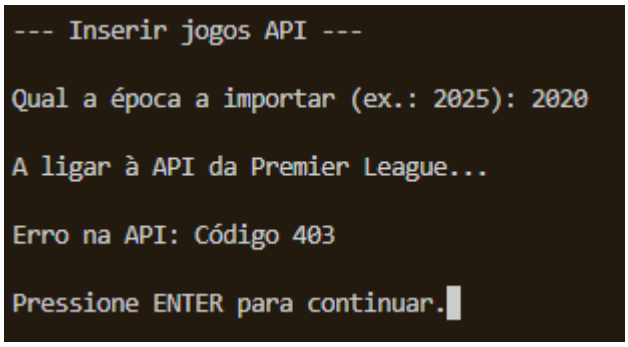


Figura 17 - Erro na resposta da API

  - **Resultado Obtido:** O sistema apresentou a mensagem: "*Sucesso! 380 jogos importados.*". Uma consulta posterior

confirma que os nomes das equipas (ex: "Manchester City FC") e resultados foram carregados corretamente.

3. Teste de Persistência (Gravar e Carregar CSV)

Objetivo: Garantir que os dados não se perdem ao fechar o programa, e que são lidos corretamente ao carregar ficheiro.

- **Cenário:**
    - **Passo 1:** Com 3 jogos na memória, selecionar opção 3 ("Guardar em Ficheiro").
    - **Passo 2:** Fechar e reabrir o programa (memória vazia).
    - **Passo 3:** Selecionar opção 4 ("Carregar do Ficheiro").
    - **Resultado Esperado:** O ficheiro jogos_premier_league.csv deve ser criado no disco e, após o carregamento, a lista deve conter exatamente os mesmos 3 jogos.
    - **Resultado Obtido:** O ficheiro foi gerado corretamente com o cabeçalho Jornada,Data,Casa,Golos C,Golos F,Fora e os dados foram recuperados na íntegra.

4. Teste de Manipulação (Pesquisar, Editar e Eliminar)

- **Cenário A: Pesquisa de Jogo**
    - **Objetivo:** Verificar a capacidade de localizar e alterar registos existentes, validando o preenchimento dos dois critérios de pesquisa obrigatórios (Jornada e Equipa).
    - **Ação:** Selecionar a opção 5 ("Editar um jogo específico").
    - **Dados de Entrada:**

        - O sistema pede a Jornada: Inserir 1.
        - O sistema pede a Equipa: Inserir Chelsea.
    - **Resultado Obtido:** O programa localiza e apresenta o jogo da Jornada 1: *Arsenal 3 - 0 Chelsea*.

- **Cenário B: Eliminação de Jogo**
    - **Objetivo:** Verificar a capacidade de localizar um jogo e eliminar o jogo encontrado.
    - **Ação:** Aceder ao Menu Eliminar (Opção 7) e escolher "2. Eliminar jogo específico".
    - **Dados de Entrada:**
        - Igual ao realizado no ponto 4. Cenário A.
    - **Resultado Esperado:** O sistema pede confirmação ("Tem a certeza...?"). Ao confirmar com 'S', o jogo é removido.
    - **Resultado Obtido:** Sucesso. Ao consultar novamente a classificação ou a lista de jogos, o jogo *Arsenal vs Chelsea* já não consta na base de dados.

- **Cenário C: Edição de jogo (edição total)**
    - **Objetivo:** Verificar a função de alteração, testando o modo de funcionamento de edição total. Este teste valida a opção onde o utilizador pretende reescrever todos os dados de um jogo (caminho alterar_tudo == 's').
    - **Pré-condição:** Existe o jogo *Jornada 1 | Arsenal 0 - 0 Chelsea*.
    - **Ação:** Selecionar "5. Editar um jogo específico".
    - **Passo 1 (Pesquisa):**

        - **Jornada: 1**
        - **Equipa: Arsenal**
    - ***Sistema:*** Encontra o jogo e pergunta: *"Pretende editar todos os campos? (S/N)"*.
        - **Passo 2:** Inserir S.
        - **Passo 3:** O sistema força a introdução de novos dados para tudo.
            - **Nova Jornada:** 2
            - **Nova Data:** 2025-05-01
            - **Nova Equipa Casa:** Tottenham
            - **Nova Equipa Fora:** Fulham
            - **Novos Golos:** 2 e 2
    - **Resultado Esperado:** O registo original (Arsenal vs Chelsea) desaparece e é substituído pelo novo registo (Tottenham vs Fulham) na lista.
    - **Resultado Obtido:** Sucesso. A listagem confirmou a alteração integral dos dados.

- **Cenário C: Edição de jogo (edição parcial)**
    - **Objetivo:** Verificar a função de alteração, testando o modo de funcionamento de edição parcial. Este teste valida a opção onde o utilizador pretende reescrever apenas certos dados de um jogo (caminho alterar_tudo == 'n').
    - **Pré-condição:** Existe o jogo Jornada 5 | Liverpool 1 - 1 Man City.

- **Ação:** Selecionar "5. Editar um jogo específico".
  - **Passo 1 (Pesquisa):**
    - **Jornada:** 5
    - **Equipa:** Liverpool
    - **Sistema:** Encontra o jogo e pergunta: "Pretende editar todos os campos? (S/N)".
  - **Passo 2:** Inserir S.
  - **Passo 3 (Edição Seletiva):**
    - **Jornada atual (5). Nova jornada:** Pressionar ENTER (mantém 5).
    - **Data atual (...). Nova data:** Pressionar ENTER (mantém data).
    - **Equipa Casa atual (Liverpool). Nova Equipa:** Pressionar ENTER (mantém Liverpool).
    - **Golos Liverpool atuais (1). Novos golos:** Inserir 3 (altera valor).
    - **Equipa Fora atual (Man City). Nova equipa:** Pressionar ENTER (mantém Man City).
    - **Golos Man City atuais (1). Novos golos:** Pressionar ENTER (mantém 1).
- **Resultado Esperado:** O jogo mantém-se na Jornada 5, com as mesmas equipas e data, mas o resultado passa a ser 3 - 1.
- **Resultado Obtido: Sucesso.** O sistema atualizou apenas o campo modificado.

5. Teste de Estatísticas e Classificação

Objetivo: Validar a lógica matemática dos cálculos.

Utilizaram-se os seguintes dados de teste controlados:

1. **Jogo 1:** Arsenal **3 - 0** Chelsea
2. **Jogo 2:** Liverpool **1 - 1** Man City

- **Teste da Tabela de Classificação:**
    - *Cálculo Manual:* Arsenal (3 pts), Man City (1 pt), Liverpool (1 pt), Chelsea (0 pts).
    - *Resultado do Programa:* A tabela gerada respeitou exatamente esta ordem e pontuação.
- **Teste da Média de Golos:**
    - *Cálculo Manual:* Total de golos = 3 + 0 + 1 + 1 = 5 golos. Total jogos = 2. Média = 2.5.
    - *Resultado do Programa:* A função media_golos_por_jogo devolveu: "*A média foi de 2.50 golos/jogo*".
- Teste da Equipa com Mais Golos Marcados:
    - Análise Manual: Arsenal (3 golos), Liverpool (1 golo), Man City (1 golo), Chelsea (0 golos).
    - Resultado do Programa: A função equipa_mais_golos identificou corretamente: "A equipa com mais golos marcados é: Arsenal (Total: 3)"
- **Teste da Pior Defesa (Mais Golos Sofridos):**

- *Análise:* O Chelsea sofreu 3 golos. As outras equipas sofreram 0 ou 1.
- *Resultado do Programa:* A função identificou corretamente: *"A equipa com mais golos SOFRIDOS é: Chelsea (Total: 3)*".

Desenvolvimento do sistema

O sistema foi desenvolvido utilizando a linguagem **Python** (versão 3.12). Para garantir a organização, a manutenibilidade e a portabilidade do projeto, foram seguidos os seguintes passos técnicos:

1. **Ambiente Virtual:** Foi criado um ambiente virtual (.venv) para isolar as dependências do projeto, garantindo que a execução não interfere com outras configurações do sistema operativo.

2. **Bibliotecas:**

    - **os**: Biblioteca padrão utilizada para melhorar a experiência do utilizador (limpeza da consola com cls) e verificação de caminhos

    - **datetime**: Utilizada para validação de datas no formato AAAA-MM-DD.

    - **requests**: Biblioteca externa (instalada via pip install requests), essencial para efetuar os pedidos HTTP à API *football-data.org*. Foi escolhida pela sua simplicidade e eficiência na gestão de pedidos GET e cabeçalhos de autenticação.

    - **tkinter**: Biblioteca de interface gráfica padrão do Python. Foi utilizado especificamente o módulo ***filedialog*** para permitir a abertura de janelas de seleção de ficheiros do sistema operativo, evitando que o utilizador tenha de digitar manualmente o nome do ficheiro CSV.

3. **Estrutura de Ficheiros:** O código foi modularizado em quatro ficheiros principais:

    - **main.py**: Ponto de entrada e gestão do menu principal.

- **dados.py**: Funções de manipulação de dados (importar API, CSV, alterar, eliminar).
- **estatistica.py**: Funções de cálculo e apresentação de dados (tabelas, médias).
- **validacoes.py**: Funções auxiliares para garantir a integridade dos inputs do utilizador.

Pesquisa e Estudo:

Durante o desenvolvimento do projeto, foi necessário realizar pesquisa técnica e estudo aprofundado sobre funções e conceitos específicos da linguagem Python para resolver problemas complexos de lógica e integração. Destacam-se os seguintes tópicos:

1. **Consumo de API REST e Formato JSON**

Para cumprir o requisito de "Gerar dados automaticamente", foi necessário estudar o funcionamento do protocolo HTTP e da biblioteca ***requests***.

- **Desafio:** Autenticar o pedido e tratar a resposta.
- **Solução:** Compreendeu-se a utilização de **headers** para enviar o *token* de autenticação (X-Auth-Token) de forma segura, em vez de o colocar no URL. Adicionalmente, estudou-se o método **.json()**, que converte automaticamente a resposta textual da API (formato JSON) em dicionários e listas de Python, permitindo a iteração direta sobre os dados dos jogos.

2. **Ordenação Personalizada com Funções Lambda**

Na elaboração da Tabela de Classificação e das estatísticas (ex: "Jogos com mais golos"), a função **sort()** padrão não era suficiente, pois

ordenava as listas pelo primeiro elemento (alfabeticamente ou pela jornada).

- **Desafio**: Ordenar uma lista de listas com base num índice específico (ex: Pontos ou Total de Golos) e de forma decrescente.

- **Solução**: Pesquisou-se sobre a função **sorted()** combinada com o **argumento key** e **funções anónimas (lambda)**.

- **Exemplo:** key=lambda x: x[1] permite instruir o Python a "olhar" apenas para o índice 1 (pontos) de cada equipa durante a ordenação.

3. Agrupamento de Dados com Dicionários

Embora existam bibliotecas, como o **pandas** que simplificariam este tipo de analise de dados, optamos por realizar com recurso a funções Python nativas. Para calcular a classificação, foi necessário transformar uma lista linear de jogos numa tabela consolidada por equipa.

- **Desafio:** Somar pontos e golos de equipas que aparecem múltiplas vezes na lista de jogos (ora em casa, ora fora).

- **Solução:** Estudou-se a estrutura de dados **Dicionário** ({chave: valor}). A lógica implementada verifica se a equipa já existe como "chave"; se não existir, cria uma entrada inicial ([0, 0]); se existir, acumula os novos valores. Esta abordagem revelou-se muito mais eficiente do que utilizar múltiplos ciclos ou listas paralelas.

4. **Manipulação de Ficheiros de Texto (Parsing Manual)**

Embora existam bibliotecas para CSV, optou-se por implementar a lógica manualmente para demonstrar domínio sobre manipulação de strings.

- **Funções estudadas:**
    - **.strip():** Fundamental para limpar caracteres invisíveis (como quebras de linha \n) que causavam erros na conversão de números.
    - **.split(','):** Utilizada para partir cada linha do ficheiro de texto e transformá-la numa lista de atributos, permitindo reconstruir os dados do jogo em memória.

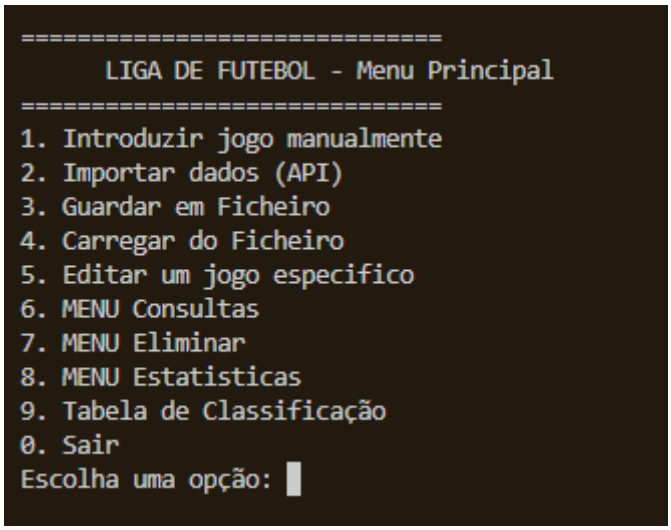
5. **Interface Gráfica para Seleção de Ficheiros (Melhoria de UX)**

Como funcionalidade de valorização do projeto e para melhorar a experiência de utilização (UX), introduziu-se na função carregar_dados_csv o uso da biblioteca **tkinter**.

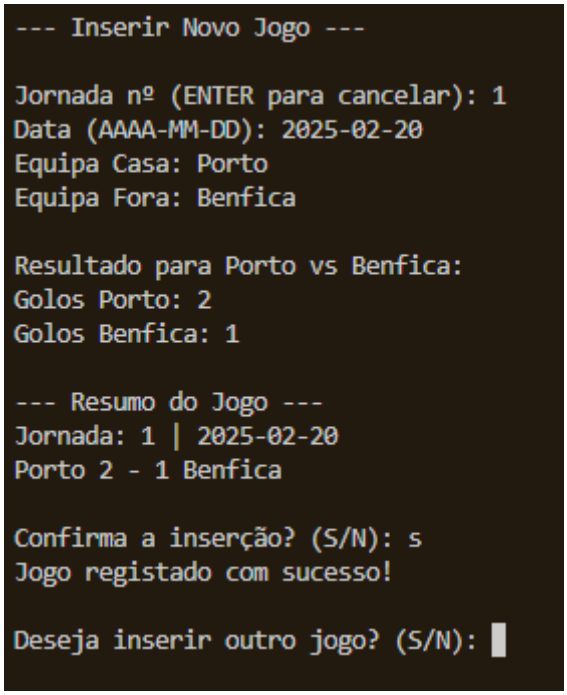
- **Desafio:** A introdução manual do nome ou caminho do ficheiro (ex: digitar "jogos.csv") é suscetível a erros de escrita e obriga o utilizador a saber o nome exato do ficheiro de memória.
- **Solução:** Implementou-se o método filedialog.askopenfilename, que invoca a janela de exploração de ficheiros nativa do sistema operativo (Windows Explorer ou Finder). Isto permite ao utilizador navegar visualmente pelas pastas e selecionar o ficheiro desejado. Adicionalmente, utilizou-se o comando root.withdraw() para ocultar a janela principal da interface gráfica, mantendo a coerência visual da aplicação que corre na consola.

Programa em imagens

Menu Principal:



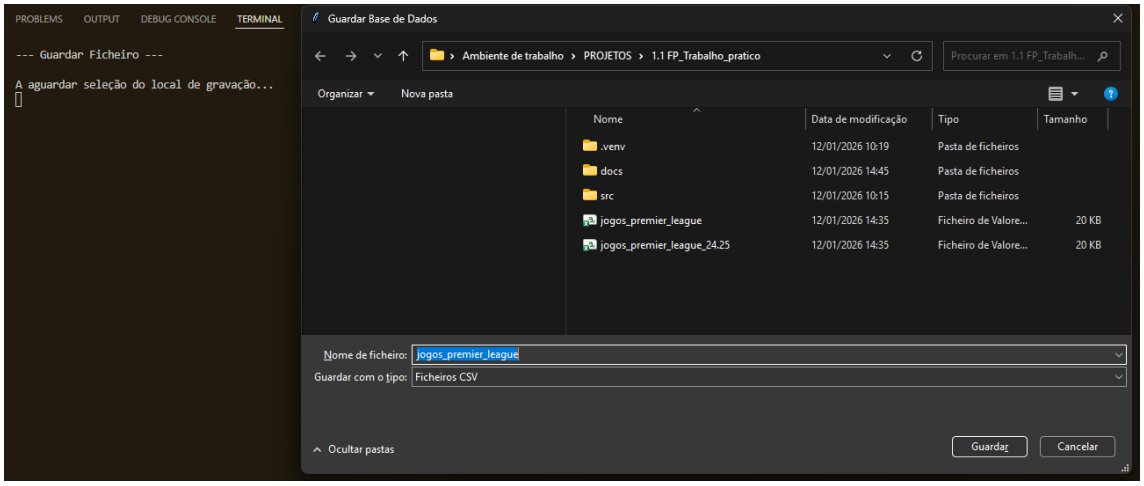
Inserir Novo Jogo:



Inserir dados (API):

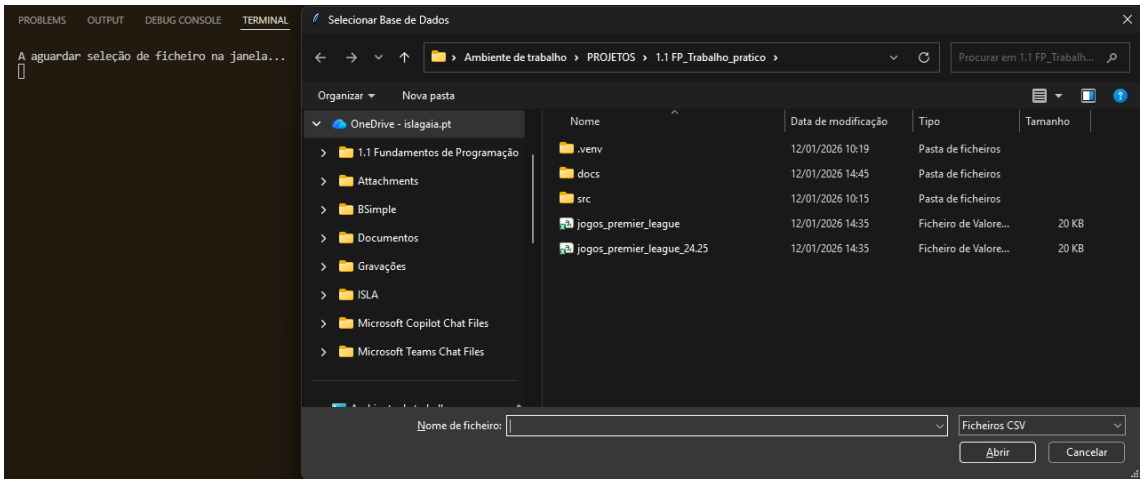
```
--- Inserir jogos API ---

Qual a época a importar (ex.: 2025): 2023

A ligar à API da Premier League...

Sucesso! 380 jogos importados.

Pressione ENTER para continuar.
```

Guardar ficheiro .CSV:




```
--- Guardar Ficheiro ---

A aguardar seleção do local de gravação...
Ficheiro 'C:/Users/joaor/Desktop/PROJETOS/1.1 FP_Trabalho_pratico/jogos_premier_league.csv' gravado com sucesso!

Primir ENTER para continuar.
```

Carregar ficheiro .CSV:




```
A aguardar seleção de ficheiro na janela...
------------------------------
Nome do ficheiro: C:/Users/joaor/Desktop/PROJETOS/1.1 FP_Trabalho_pratico/jogos_premier_league_24.25.csv carregado com sucesso.
380 jogos na memória.
Pressione Enter para continuar...
```

Editar um jogo específico:

```
--- Pesquisar Jogo ---

Qual a Jornada que procura (1-38)? (ENTER para sair) 1

Qual a Equipa (Casa ou Fora)? ars

A procurar na Jornada 1 por 'ars'...


ENCONTRADO: J1 | 2024-08-17 | Arsenal FC 2 - 0 Wolverhampton Wanderers FC

Pressione ENTER para continuar...

--- Menu de Edição ---
Tem a certeza que quer alterar este jogo? (S/N): s

Como deseja editar?
S - Editar TUDO (Reescrever todos os dados do jogo)
N - Editar Campo a Campo (Para manter valor, primir ENTER)
Opção (S/N):
```

Sub-Menu Consultas:

```
============================
      LIGA DE FUTEBOL - Menu de Consulta
============================
1. Consultar todos os jogos
2. Consultar jogo especifico
0. Voltar ao menu principal
Escolha uma opção:
```

Consultar todos os jogos:

```
--- Consultar Todos os Jogos ---

--- NOTA: jogos por realizar tem "golos" = -1


--- Jornada 1 ---
  2024-08-16 - Manchester United FC vs Fulham FC (1-0)
  2024-08-17 - Ipswich Town FC vs Liverpool FC (0-2)
  2024-08-17 - Arsenal FC vs Wolverhampton Wanderers FC (2-0)
  2024-08-17 - Everton FC vs Brighton & Hove Albion FC (0-3)
  2024-08-17 - Newcastle United FC vs Southampton FC (1-0)
  2024-08-17 - Nottingham Forest FC vs AFC Bournemouth (1-1)
  2024-08-17 - West Ham United FC vs Aston Villa FC (1-2)
  2024-08-18 - Brentford FC vs Crystal Palace FC (2-1)
  2024-08-18 - Chelsea FC vs Manchester City FC (0-2)
  2024-08-19 - Leicester City FC vs Tottenham Hotspur FC (1-1)

--- Jornada 2 ---
  2024-08-24 - Brighton & Hove Albion FC vs Manchester United FC (2-1)
  2024-08-24 - Crystal Palace FC vs West Ham United FC (0-2)
  2024-08-24 - Fulham FC vs Leicester City FC (2-1)
  2024-08-24 - Manchester City FC vs Ipswich Town FC (4-1)
  2024-08-24 - Southampton FC vs Nottingham Forest FC (0-1)
  2024-08-24 - Tottenham Hotspur FC vs Everton FC (4-0)
  2024-08-24 - Aston Villa FC vs Arsenal FC (0-2)
  2024-08-25 - AFC Bournemouth vs Newcastle United FC (1-1)
  2024-08-25 - Wolverhampton Wanderers FC vs Chelsea FC (2-6)
  2024-08-25 - Liverpool FC vs Brentford FC (2-0)
```

Consultar jogo específico:

```
--- Pesquisar Jogo ---

Qual a Jornada que procura (1-38)? (ENTER para sair) 1

Qual a Equipa (Casa ou Fora)? ARs

A procurar na Jornada 1 por 'ars'...


ENCONTRADO: J1 | 2024-08-17 | Arsenal FC 2 - 0 Wolverhampton Wanderers FC

Pressione ENTER para continuar...
```

Sub-Menu Eliminar:

```
==============================
      LIGA DE FUTEBOL - Menu de Eliminar
==============================
1. Eliminar todos os jogos
2. Eliminar jogo especifico
0. Voltar ao menu principal
Escolha uma opção:
```

Eliminar todos os jogos:

```
--- ATENÇÃO: Eliminar Todos os Jogos ---

Tem a certeza que quer apagar TODOS os jogos? (S/N):
```

Eliminar jogo específico:

```
--- Pesquisar Jogo ---

Qual a Jornada que procura (1-38)? (ENTER para sair) 1

Qual a Equipa (Casa ou Fora)? ars

A procurar na Jornada 1 por 'ars'...


ENCONTRADO: J1 | 2024-08-17 | Arsenal FC 2 - 0 Wolverhampton Wanderers FC

Pressione ENTER para continuar...

--- APAGAR Jogo ---

Tem a certeza que quer APAGAR este jogo? (S/N):
```

Sub-Menu Estatísticas:

```
==============================
      LIGA DE FUTEBOL - Menu de Estatisticas
==============================
1. Jogos com mais golos
2. Média de golos por jogo
3. Equipa com mais golos marcados
4. Equipa com mais golos sofridos
0. Voltar ao menu principal
Escolha uma opção:
```

Jogos com mais golos:

```
Top 10 jogos com mais golos:

Jornada 17 | 2024-12-22 | Tottenham Hotspur FC 3 - 6 Liverpool FC  (Total: 9)
Jornada 2 | 2024-08-25 | Wolverhampton Wanderers FC 2 - 6 Chelsea FC  (Total: 8)
Jornada 7 | 2024-10-05 | Brentford FC 5 - 3 Wolverhampton Wanderers FC  (Total: 8)
Jornada 9 | 2024-10-26 | Brentford FC 4 - 3 Ipswich Town FC  (Total: 7)
Jornada 13 | 2024-11-30 | West Ham United FC 2 - 5 Arsenal FC  (Total: 7)
Jornada 15 | 2024-12-08 | Tottenham Hotspur FC 3 - 4 Chelsea FC  (Total: 7)
Jornada 24 | 2025-02-01 | Nottingham Forest FC 7 - 0 Brighton & Hove Albion FC  (Total: 7)
Jornada 26 | 2025-02-23 | Newcastle United FC 4 - 3 Nottingham Forest FC  (Total: 7)
Jornada 32 | 2025-04-12 | Manchester City FC 5 - 2 Crystal Palace FC  (Total: 7)
Jornada 35 | 2025-05-04 | Brentford FC 4 - 3 Manchester United FC  (Total: 7)

Pressione ENTER para continuar.
```

Média de golos por jogo:

```
A média foi de, 2.93 golos/jogo

Pressione ENTER para continuar.
```

Equipa com mais golos marcados:

```
A equipa com mais golos marcados é: Liverpool FC
Total de golos: 86

Pressione ENTER para continuar.
```

Equipa com mais golos sofridos:

```
A equipa com mais golos SOFRIDOS é: Southampton FC
Total de golos sofridos: 86

Pressione ENTER para continuar.
```

Tabela de Classificação:

```
POS  EQUIPA                          PTS   J
=============================================
1    liverpool fc                    84    38
2    arsenal fc                      74    38
3    manchester city fc              71    38
4    chelsea fc                      69    38
5    newcastle united fc             66    38
6    aston villa fc                  66    38
7    nottingham forest fc            65    38
8    brighton & hove albion fc       61    38
9    afc bournemouth                 56    38
10   brentford fc                    56    38
11   fulham fc                       54    38
12   crystal palace fc               53    38
13   everton fc                      48    38
14   west ham united fc              43    38
15   manchester united fc            42    38
16   wolverhampton wanderers fc      42    38
17   tottenham hotspur fc            38    38
18   leicester city fc               25    38
19   ipswich town fc                 22    38
20   southampton fc                  12    38
=============================================

Pressione ENTER para voltar.
```

Codificação Python

Ficheiro: main.py

```
import dados
import estatistica
import validacoes

chave_api = '909f97df86f1442fa9611cc74f773bc2'

# --- Função MENU PRINCIPAL---
def main():

    lista_jogos = []  # Lista para armazenar os jogos na memória.

    while True:
        validacoes.limpar_ecra()
        print("\n" + "="*30)
        print("      LIGA DE FUTEBOL - Menu Principal     ")
        print("="*30)
        print("1. Introduzir jogo manualmente")
        print("2. Importar dados (API)")
        print("3. Guardar em Ficheiro")
        print("4. Carregar do Ficheiro")
        print("5. Editar um jogo especifico")
        print("6. MENU Consultas")
        print("7. MENU Eliminar")
        print("8. MENU Estatisticas")
        print("9. Tabela de Classificação")
        print("0. Sair")

        try:
            opcao = int(input("Escolha uma opção: "))
        except:
            print("Opção inválida!")
            input()
        else:
            if opcao == 1:
                lista_jogos = dados.inserir_jogo_manual(lista_jogos)
            elif opcao == 2:
                lista_jogos = dados.importar_jogos_api(chave_api)
            elif opcao == 3:
                dados.guardar_dados_csv("jogos_premier_league.csv", lista_jogos)
            elif opcao == 4:
                lista_jogos = dados.carregar_dados_csv()
            elif opcao == 5:
```

```
                lista_jogos = dados.alterar_jogo(lista_jogos)
            elif opcao == 6:
                dados.menu_consultar(lista_jogos)
            elif opcao == 7:
                dados.menu_eliminar(lista_jogos)
            elif opcao == 8:
                estatistica.menu_estatisticas(lista_jogos)
            elif opcao == 9:
                estatistica.tabela_classificacao(lista_jogos)
            elif opcao == 0:
                print("Adeus.")
                break
            else:
                print("Opção errada!")

# --- Fim Função MENU PRINCIPAL---

main()
```

Ficheiro: validacoes.py

```
# Ficheiro com as funções de validação de campos

import os
from datetime import datetime

# --- Função Auxiliar: LIMPAR ECRÃ ---
def limpar_ecra():
    os.system('cls' if os.name == 'nt' else 'clear')

# --- Função VALIDAR NUMERO INTEIRO
# --- Pede um número repetidamente até o utilizador inserir um valor inteiro válido.
# --- vamos inserir dois parametros MIN e MAX para ele também validar sempre que
precisarmos (como no caso da pesquisa ou inserir Jornada)
def ler_inteiro (mensagem_input, min= None, max = None, permite_vazio = False):

    while True:

        valor_testar = input(mensagem_input)

        if valor_testar == '' and permite_vazio == True:
            return None

        try:
            valor_int = int(valor_testar)

              if (min is not None and valor_int < min) or (max is not None
and valor_int > max):
                print(f"Erro: O valor tem de estar entre {min} e {max}.\n")

            else:
                return valor_int

        except:
            print("Erro: Tens de introduzir um número inteiro válido.\n")


# --- Função VALIDAR TEXTO
# --- Pede um texto e garante que não está vazio.
def ler_texto(mensagem_input, permite_vazio = False):
    while True:
        texto = input(mensagem_input).strip() # .strip() remove espaços extra

        if texto == '' and permite_vazio == True:
            return None

        elif len(texto) > 0:
```

```
            return texto

        else:
            print("Erro: Este campo não pode ficar vazio.\n")

# --- Função VALIDAÇÃO DA DATA
# --- Pede uma data e verifica se é válida no formato AAAA-MM-DD.
def ler_data(mensagem_input, permite_vazio = False):

    while True:

        valor_testar = input(mensagem_input)

        if valor_testar == '' and permite_vazio == True:
            return None

        try:
            valor_testar = valor_testar
            # Tenta converter o texto numa data real
            datetime.strptime(valor_testar, '%Y-%m-%d')
            return valor_testar

        except ValueError:
            print("Erro: Data inválida! Usa o formato AAAA-MM-DD (ex: 2026-01-
30).\n")
```

Ficheiro: dados.py

```
import os
import requests # https://requests.readthedocs.io/en/latest/
import validacoes

# necessário para abrir janela de seleção de ficheiro a carregar
import tkinter as tk
from tkinter import filedialog


'''--- Função IMPORTAR JOGOS API ---
--- Importar dados da API: https://api.football-data.org/v4/competitions/PL/matches
(PL = Premier League)

O Algoritmo da Importação:
1 - Parametros: AKI Key, chave de acesso à API; Epoca, que queremos importar
2 - Criar a função GET URL e Chave
3 - Verificar se recebemos resposta da API
    3.a - Exceção mostra ERRO com código de estado
4 - Colocamos a resposta atraves de decodificadore JSON num ficheiro
5 - Vamos ler o JSON com um ciclo, a percorrer todos os 'match'
6 - Em cada ciclo vamos ler os atributos relevantes: Jornada, Data, Equipa da Casa,
Equipa Visitante, golos casa e golos fora.
5 - Criamos a linha a inserir no CSV, e adicionamos ao CSV
6 - Retona a lista de jogos em formato CSV
'''
def importar_jogos_api(api_key):

    validacoes.limpar_ecra()
    print("--- Inserir jogos API ---\n")
    epoca = int(input("Qual a época a importar (ex.: 2025): "))

    # Adicionámos ?season=2025 no fim
    url = f"https://api.football-data.org/v4/competitions/PL/matches?season={epoca}"
    headers = { 'X-Auth-Token': api_key }    # Chave da API

    print("\nA ligar à API da Premier League...")
    resposta = requests.get(url, headers=headers)                    # função GET,
passamos os parametros URL e HEADERS

    if resposta.status_code == 200:
        dados_json = resposta.json()    # builtin JSON decoder. Se façhar o decode,
gera uma excessão (https://requests.readthedocs.io/en/latest/user/quickstart/)
        lista_jogos = []

        for jogo in dados_json['matches']:
            # vamos extrair do ficheiro json os dados do JOGO
```

```
            jornada = jogo['matchday']
            data = jogo['utcDate'][0:10] # Apenas a data (AAAA-MM-DD)
            casa = jogo['homeTeam']['name']
            fora = jogo['awayTeam']['name']

            # Agora vampos extrair os golos do jogoa (recebmos None é para jogos
ainda não realizados, atribuimos o valor "-1")
            g_casa = jogo['score']['fullTime']['home']
            g_fora = jogo['score']['fullTime']['away']

            if g_casa is None:
                g_casa = -1
                g_fora = -1

           # Cria a linha: [Jornada, Data, Casa, G_Casa, G_Fora, Fora]
            linha = [jornada, data, casa, g_casa, g_fora, fora]
            lista_jogos.append(linha)

        print(f"\nSucesso! {len(lista_jogos)} jogos importados.")
        input("\nPressione ENTER para continuar.")
        return lista_jogos
    else:
        print(f"\nErro na API: Código {resposta.status_code}")
        input("\nPressione ENTER para continuar.")
        return []
'''--- Função GUARDA CSV ---
O Algoritmo da Guardar Ficheiro:

1 - Parametros: nome do ficheiro, que se pretende gravar; lista de dados que temos
atualmente carregada no programa
2 - Tentar criar o ficheiro, em modo escrita (w)
    2.1 - Exceção, se não for possivel criar o ficheiro, apresenta erro.
3 - Escrever no ficheiro o cabeçalho, que está de acordo com a ordem definida na
função _importar_jogos_api_
4 - Ciclo para percorrer a lista de jogos e inseir linha a linha no CSV
5 - Fechar o ficheiro (gravar)
'''
def guardar_dados_csv(nome_sugerido, lista_dados):
    validacoes.limpar_ecra()
    print("--- Guardar Ficheiro ---\n")

    if lista_dados == []:
            print("\nAviso: Não existem dados na memória para guardar.")
            input("\nPressione ENTER para continuar...")
            return

    # Abrir Janela de "Guardar Como"
    print("A aguardar seleção do local de gravação...")
```

```
    root = tk.Tk()
    root.withdraw() # Esconde a janela principal cinzenta

    # Abre a janela de salvar, sugerindo o nome que vem do main.py
    nome_ficheiro = filedialog.asksaveasfilename(
        title="Guardar Base de Dados",
        initialfile=nome_sugerido,       # Sugere "jogos_premier_league.csv"
        defaultextension=".csv",         # Se o utilizador não escrever extensão,
mete .csv
        filetypes=[("Ficheiros CSV", "*.csv"), ("Todos os ficheiros", "*.*")]
    )

    root.destroy() # Fecha o processo da janela


    # Recebe a lista de jogos e guarda num ficheiro de texto (CSV).
    # Criar com excessão
    try:
        # Abrir o ficheiro em modo de escrita ('w')
        ficheiro = open(nome_ficheiro, 'w', newline='', encoding='utf-8')

        # Escrever o cabeçalho
        ficheiro.write("Jornada,Data,Casa,Golos C,Golos F,Fora\n")

        # Agora vamos percorrer a lista de jogos do nosso CSV
        for jogo in lista_dados:
            # cada jogo é uma lista: [1, '2025-08...', 'Arsenal', 2, 0, 'Chelsea']
conforme o cabeçalho que definimos.
            linha_texto = str(jogo[0]) + "," + str(jogo[1]) + "," + jogo[2] + "," +
str(jogo[3]) + "," + str(jogo[4]) + "," + jogo[5] + "\n"
            # agora temos a linha da forma que pretendemos escrever no ficheiro
            ficheiro.write(linha_texto)

        # Fechar o ficheiro
        ficheiro.close()
        print(f"Ficheiro '{nome_ficheiro}' gravado com sucesso!")

    except:
        print(f"Erro ao gravar ficheiro")

    print(input('\nPrimir ENTER para continuar.'))

'''--- Função CARREGAR DO FICHEIRO CSV ---
O Algoritmo da Carregar dados de um Ficheiro:
------------------ Podemos criar aqui janela de ABRIR FICHEIRO----------------------
--------------------- !!!!!!!!!!!!!!!!!!
1 - Parametros: nome do ficheiro que queremos abrir
2 - Tentar abrir o ficheiro, modo leitura
    2.1 - Exceção, se não for possivel ler o ficheiro, apresenta erro.
```

```
3 - Ler todas as linhas, uma por uma (funão readlines) para uma lista
4 - Fechar o ficheiro
5 - Ciclo de ler cada linha da lista criada (saltar linha do cabeçalho)
6 - A cada linha (lista) da lista de linhas, separar os dados e carregar no indice
correto de acordo com o definido na função _importar_jogos_api_
7 - Cada linha, é adicionada à lista de jogos
8 - Retona a lista (de listas) de jogos
'''
def carregar_dados_csv(nome_ficheiro=None):

    validacoes.limpar_ecra()

    # SE não foi passado nome, abrimos a janela (código fonte:
https://gemini.google.com/app/0dd27cfbbf71be87 )
    if nome_ficheiro is None:
        print("A aguardar seleção de ficheiro na janela...")
        root = tk.Tk()
        root.withdraw() # Esconde a janela principal "feia" do Tkinter

        # Abre o explorador de ficheiros
        nome_ficheiro = filedialog.askopenfilename(
            title="Selecionar Base de Dados",
            filetypes=[("Ficheiros CSV", "*.csv"), ("Todos os ficheiros", "*.*")]
        )

        root.destroy() # Fecha a instância do Tkinter

    # Se o utilizador cancelou a janela ou string vazia
    if not nome_ficheiro:
        print("Nenhum ficheiro selecionado.")
        input("\nPressione ENTER para continuar...")
        return []

    #     Lê o ficheiro e separa as palavras pelas vírgulas (split).
    lista_jogos = []

    try:
        # Abrir para leitura ('r')
        ficheiro = open(nome_ficheiro, 'r', encoding='utf-8')

        # Ler todas as linhas para uma lista de strings
        linhas_do_ficheiro = ficheiro.readlines()

        # Fechar ficheiro
        ficheiro.close()

        # Ler cada linha, e começamos no índice 1 para saltar o cabeçalho (que está
no índice 0)
```

```
        for i in range(1, len(linhas_do_ficheiro)): # range de 1 ao comprimento da
nossa lista de listas

            linha_atual = linhas_do_ficheiro[i]

            # .strip() remove o '\n' (o enter) do final da linha
            linha_limpa = linha_atual.rstrip()

            # .split(',') parte a frase onde houver vírgulas e cria uma lista com
esses valores
            dados_linha = linha_limpa.split(',')

            # Agora temos uma lista de strings, ex: ['1', '2025...', 'Arsenal'...]
            # Precisamos de converter os valores para números inteiros

            jornada = int(dados_linha[0])
            data = dados_linha[1]
            casa = dados_linha[2]
            g_casa = int(dados_linha[3])
            g_fora = int(dados_linha[4])
            fora = dados_linha[5]

            # Reconstruir a lista final
            jogo_formatado = [jornada, data, casa, g_casa, g_fora, fora]
            lista_jogos.append(jogo_formatado)

        print("-"*30)
        print(f'Nome do ficheiro: {nome_ficheiro} carregado com sucesso.')
        print(f"{len(lista_jogos)} jogos na memória.")
        print(input("Pressione Enter para continuar..."))
        return lista_jogos

    except:
        print(f"Erro ao ler ficheiro.")
        input("\nPressione ENTER para continuar...")
        return []

''' --- Função INSERIR JOGOS MANUALMENTE ---
Algoritmo da Inserção Manual:
1 - Parametro: lista de listas de jogos existente
2 - Tentar pedir um por um os atributos de cada jogo
3 - Criar a lista do jogo, com os atributos inseridos
-----------------------------------------------------  CRIAR VALIDAÇÃO
JORNADA+EQUIPA, não aceitar duplicado  ----------------------------
4 - Adicionar o jogo à lista de jogos

'''
'''def inserir_jogo_manual(lista_jogos):
    # Ciclo infinito para permitir inserir vários jogos
```

```
    while True:
        validacoes.limpar_ecra()
        # Pede ao utilizador os dados de um jogo e adicionar jogo (lista) à lista de
jogos (lista de listas).
        print("\n--- Inserir Novo Jogo ---")
        try:
            # Pedir os dados um a um, mas permitir cancelar ação antes de inserir
jornada (ENTER para sair)
            jornada = validacoes.ler_inteiro('Jornada nº (ENTER para cancelar): ', 1,
38, permite_vazio =True)

            if jornada is None:
                print("Operação cancelada.\n")
                # E quer continuar a inserir outro jogo?
                continuar = input("Deseja inserir outro jogo? (S/N):
").strip().lower()
                if continuar != 's':
                    return lista_jogos  # Sai do ciclo infinito e retorna a lista
atualizada
                else:
                    continue  # volta ao inicio do ciclo para inserir novo jogo

            data = validacoes.ler_data("Data (AAAA-MM-DD): ")
            casa = validacoes.ler_texto("Equipa Casa: ")
            fora = validacoes.ler_texto("Equipa Fora: ")

            print(f"Insira o resultado para {casa} vs {fora}:")
            g_casa = validacoes.ler_inteiro(f"Golos {casa}: ", 0, 20)
            g_fora = validacoes.ler_inteiro(f"Golos {fora}: ", 0, 20)

            # Criar a linha igual à estrutura da API: [Jornada, Data, Casa, GC, GF,
Fora]
            novo_jogo = [jornada, data, casa, g_casa, g_fora, fora]

            # Confirmar dados de inserção e se pretende adicionar ou cancelar
            print("\nNovo Jogo a Registar:")
            print(f"Jornada: {jornada}, Data: {data}, {casa} {g_casa} - {g_fora}
{fora}\n")

            confirmar = input("Confirma a inserção deste jogo? (S/N):
").strip().lower()
            if confirmar != 's':
                print("Operação cancelada.\n")
                # E quer continuar a inserir outro jogo?
                continuar = input("Deseja inserir outro jogo? (S/N):
").strip().lower()
                if continuar != 's':
                    return lista_jogos  # Sai do ciclo infinito e retorna a lista
atualizada
```

```
                else:
                    continue  # volta ao inicio do ciclo para inserir novo jogo

            # Adicionar à lista principal
            lista_jogos.append(novo_jogo)
            print("Jogo registado com sucesso!\n")
            consultar_todos_jogos(lista_jogos)
            print(input(''))

            # Perguntar se quer inserir outro jogo
            continuar = input("Deseja inserir outro jogo? (S/N): ").strip().lower()
            if continuar != 's':
                return lista_jogos  # Sai do ciclo infinito e retorna a lista
atualizada

        except:
            print("ERRO: Não foi possivel registar o jogo.")
            continuar = input("Deseja inserir outro jogo? (S/N): ").strip().lower()
            if continuar != 's':
                return lista_jogos # Sai do ciclo infinito e retorna a lista
atualizada
'''
# --- Função INSERIR JOGOS MANUALMENTE ---
def inserir_jogo_manual(lista_jogos):
    while True:
        validacoes.limpar_ecra()
        print("--- Inserir Novo Jogo ---\n")

        jornada = validacoes.ler_inteiro('Jornada nº (ENTER para cancelar): ', 1, 38,
permite_vazio=True)
        if jornada is None:
            return lista_jogos

        data = validacoes.ler_data("Data (AAAA-MM-DD): ")
        casa = validacoes.ler_texto("Equipa Casa: ")
        fora = validacoes.ler_texto("Equipa Fora: ")

        print(f"\nResultado para {casa} vs {fora}:")
        g_casa = validacoes.ler_inteiro(f"Golos {casa}: ", 0, 20)
        g_fora = validacoes.ler_inteiro(f"Golos {fora}: ", 0, 20)

        novo_jogo = [jornada, data, casa, g_casa, g_fora, fora]

        print("\n--- Resumo do Jogo ---")
        print(f"Jornada: {jornada} | {data}")
        print(f"{casa} {g_casa} - {g_fora} {fora}")

        confirmar = input("\nConfirma a inserção? (S/N): ").strip().lower()
        if confirmar == 's':
```

```
            lista_jogos.append(novo_jogo)
            print("Jogo registado com sucesso!")
        else:
            print("Operação cancelada.")
            validacoes.limpar_ecra()

        if input("\nDeseja inserir outro jogo? (S/N): ").strip().lower() != 's':
            return lista_jogos


'''--- Funcao ELIMINAR TUDO ---
O Algoritmo da Eliminação de todos os jogos:
1 - Parametro: Lista de jogos
2 - Mensagem de confirmação se pretende de facto eliminar tudo
    2.a - Se "N", sistema não elimina e mostra mensagem "Opoeração Cancelada"
3 - Se "S", sistema vai limpar a variavel global lista de jogos
(Nota: não precisa de return por ser uma variavel global)
'''
def eliminar_tudo(lista_jogos):
    validacoes.limpar_ecra()
    print("--- ATENÇÃO: Eliminar Todos os Jogos ---\n")

    confirmar = input("Tem a certeza que quer apagar TODOS os jogos? (S/N): ")
    if confirmar.lower() == 's':
        lista_jogos # Precisamos de global para "limpar" a variavel original ---
PRECISO DISTO AQUI??
        lista_jogos.clear()  # Limpa a lista de jogos
        print("Todos os jogos foram apagados da memória.")

    else:
        print("Operação cancelada.")

    input("Pressione Enter para continuar...")

''' --- Funcao PESQUISAR JOGO ESPECIFICO ---
Algoritmo da Pesquisa de um jogo especifico:
1 - Parametro: Lista de jogos
2 - Pede a jornada que se pretende pesquisar (Invoca a função de validação, para
validação inteiros, texto e data)
    2.a Opção '0' permite sair da pesquisa
3 - Pede equipa a pesquisar (que pode ter jogado em casa ou fora)
4 - Sistema vai iterar a lista de jogos, e para cada jogos separa os atributos (da
lista do jogo)) e compara com os valores pesquisados.
5 - Se encontrar valor igual para jornada E equipa, apresenta o jogo encontrado
    5.a - Se não encontrar jornada E equipa, mostra mensagem de erro
6 - Apos mostrar permitem ao utilizador repetir a pesquisa
7 - A função retorna sempre o indice do jogo (referente à lista de jogos) para poder
ser usada para Alterar e Eliminar jogo.
    7.a - Ou no caso da pesquisa não ter resultados não devolve indice
```

```
'''
def pesquisar_jogo(lista_jogos):
    validacoes.limpar_ecra()
    print("--- Pesquisar Jogo ---")

    while True:

        jornada_pesquisar = validacoes.ler_inteiro("\nQual a Jornada que procura (1-
38)? (ENTER para sair) ", 1, 38, permite_vazio = True)

        if jornada_pesquisar is None:
            print('Sair da pesquisa')
            return -1, None

        # Pedir a equipa a pesquisar
        equipa_pesquisar = validacoes.ler_texto("\nQual a Equipa (Casa ou Fora)?
").lower()

        print(f"\nA procurar na Jornada {jornada_pesquisar} por
'{equipa_pesquisar}'...\n")

        for i in range(len(lista_jogos)):
            jogo = lista_jogos[i]
            # jogo: [0=Jornada, 1=Data, 2=Casa, 3=GC, 4=GF, 5=Fora]

            if jogo[0] == jornada_pesquisar and (equipa_pesquisar in jogo[2].lower()
or equipa_pesquisar in jogo[5].lower()):
                resumo = f"ENCONTRADO: J{jogo[0]} | {jogo[1]} | {jogo[2]} {jogo[3]} -
{jogo[4]} {jogo[5]}"
                print("\n" + resumo)
                input("\nPressione ENTER para continuar...")
                return i, resumo

        print("\nJogo não encontrado.")
        input("\nPressione ENTER para continuar...")
        return -1, None

''' --- Funcao ALTERAR JOGO PESQUISADO ---
O Algoritmo da Eliminação:
1 - Parametro: Lista de jogos
2 - Pedir Jornada e Equipa (usar Funcao PESQUISAR JOGO ESPECIFICO).
    2.1 - Percorrer a lista à procura.
        2.1.1 - Se não encontrar: Mostra mensagem de erro
    2.1 - Se encontrar: Mostrar o jogo ao utilizador.
4 - Pedir confirmação ("Tem a certeza que pretende eliminar o jogo?").
5 - Se sim, apagar usando .pop(i).
    5.1 - Se não, cancela a ação e voltar ao inicio da função eliminar
6 - Continuar a apagar jogos?
7 - Se sim, volta para o inicio do algoritmo
```

```
Parar (break), porque a lista mudou de tamanho e continuar o loop daria erro.
'''
def alterar_jogo(lista_jogos):

    indice_jogo_alterar, str_jogo_alterar = pesquisar_jogo(lista_jogos)

    if indice_jogo_alterar == -1:
        print("Nenhum jogo seleccionado.")
        return lista_jogos # termina aqui a função

    print("\n--- Menu de Edição ---")

    confirmar_alterar = input("Tem a certeza que quer alterar este jogo? (S/N):
").strip().lower()

    if confirmar_alterar == 's':

        print("\nComo deseja editar?")
        print("S - Editar TUDO (Reescrever todos os dados do jogo)")
        print("N - Editar Campo a Campo (Para manter valor, primir ENTER)")
        opcao = input("Opção (S/N): ").strip().lower()

        if opcao == 's':

            # Pedir os dados um a um, mas vamos obrigar a preencher novos valores em
todos eles
            jornada = validacoes.ler_inteiro("\nJornada (1-38): \n", 1, 38)
            data = validacoes.ler_data("Data (AAAA-MM-DD): \n")
            casa = validacoes.ler_texto("Equipa Casa: \n")
            fora = validacoes.ler_texto("Equipa Fora: \n")

            print(f"Insira o resultado para {casa} vs {fora}:\n")
            g_casa = validacoes.ler_inteiro(f"Golos {casa}: \n", 0, 20)
            g_fora = validacoes.ler_inteiro(f"Golos {fora}: \n", 0, 20)

            # Criar a linha igual à estrutura da API: [Jornada, Data, Casa, GC, GF,
Fora]
            jogo_alterado = [jornada, data, casa, g_casa, g_fora, fora]

            print(str_jogo_alterar)
            print(f'\nALTERADO PARA: {jogo_alterado[0]} | {jogo_alterado[1]} |
{jogo_alterado[2]}  {jogo_alterado[3]} - {jogo_alterado[4]} {jogo_alterado[5]}\n')

            # continuar a alterar depois de rever o antes e o depois
            confirmar_alteracao_final = input("Confirma a alteração deste jogo?
(S/N): ").strip().lower()
            if confirmar_alteracao_final == 's':
                # Adicionar à lista principal
```

```
                lista_jogos[indice_jogo_alterar] = jogo_alterado # no indice do jogo
alterar, colocamos o jogo alterado
                print("Jogo alterado com sucesso!")
            else:
                print("Operação cancelada.")
                return lista_jogos # termina aqui a função

        # --- Se não é todo o jogo, então temos de ir campo a campo perguntar se quer
alterar esse campo ---
        elif opcao == 'n':
            # Edição campo-a-campo, ENTER mantém valor atual

            # guardo uma copia da lista do jogo selecionado, para comparar no final
se existiram de facto alterações
            jogo_selecionado = lista_jogos[indice_jogo_alterar].copy()

            print("ENTER para manter o valor atual.")

            # Jornada (aceita ENTER para manter)
            nova_jornada = validacoes.ler_inteiro(f"Jornada atual
({jogo_selecionado[0]}). Nova jornada: ", 1, 38, True)
            if nova_jornada is not None:
                jogo_selecionado[0] = nova_jornada

            # Data
            nova_data = validacoes.ler_data(f"Data atual ({jogo_selecionado[1]}).
Nova data: ", permite_vazio=True)
            if nova_data is not None:
                jogo_selecionado[1] = nova_data

            # Equipa Casa
            nova_casa = validacoes.ler_texto(f"Equipa Casa atual
({jogo_selecionado[2]}). Nova Equipa: ", permite_vazio=True)
            if nova_casa is not None:
                jogo_selecionado[2] = nova_casa

            # Golos Casa
            novos_golos_casa = validacoes.ler_inteiro(f"Golos {jogo_selecionado[2]}
atuais ({jogo_selecionado[3]}). Novos golos: ", 0, 20, permite_vazio=True)
            if novos_golos_casa is not None:
                jogo_selecionado[3] = int(novos_golos_casa)

            # Equipa Fora
            nova_fora = validacoes.ler_texto(f"Equipa Fora atual
({jogo_selecionado[5]}). Nova equipa: ", permite_vazio=True)
            if nova_fora is not None:
                jogo_selecionado[5] = nova_fora

            # Golos Fora
```

```
            novos_golos_fora = validacoes.ler_inteiro(f"Golos {jogo_selecionado[5]}
atuais ({jogo_selecionado[4]}). Novos golos: ", 0, 20, permite_vazio=True)
            if novos_golos_fora is not None:
                jogo_selecionado[4] = int(novos_golos_fora)

            if lista_jogos[indice_jogo_alterar] == jogo_selecionado: #compara a copia
que fizesmos (original) com o que foi alterado
                print('Não realizou nenhuma alteração.')
                print(input('\nPrimir ENTER para continuar.'))
                return lista_jogos # termina aqui a função

            print(str_jogo_alterar)
            print(f'\nALTERADO PARA: {jogo_selecionado[0]} | {jogo_selecionado[1]} |
{jogo_selecionado[2]}  {jogo_selecionado[3]} - {jogo_selecionado[4]}
{jogo_selecionado[5]}\n')

            # Confirmar ação final
            confirmar_alterar = input("Confirma a alteração deste jogo? (S/N):
").strip().lower()
            if confirmar_alterar == 's':
                # Adicionar à lista principal
                lista_jogos[indice_jogo_alterar] = jogo_selecionado
                print("\nJogo alterado com sucesso!\n")
                print(input('\nPrimir ENTER para continuar.'))
            else:
                print("Alteração cancelada.")
                return lista_jogos # termina aqui a função
        else:
            print("Opção inválida.\n")
    else:
        print("Operação cancelada.\n")
    return lista_jogos

'''--- Funcao ELIMINAR JOGO PESQUISADO ---
O Algoritmo da ELIMINAR jogo pesquisado:
1 - Pedir Jornada e Equipa (Função pesquisar_jogo --» devolve indice do jogo e string
resumo). (opção 0 para sair da função)
    1.1 - Se não encontrar: Mostra mensagem de erro
2 - Se encontrar: Mostrar o jogo ao utilizador.
8 - Perguntar se quer Continuar a apagar jogos?
7 - Se sim, volta para o inicio do algoritmo (opção 0 para sair da função)
Parar (break), porque a lista mudou de tamanho e continuar o loop daria erro.
'''
def eliminar_jogo(lista_jogos):
    indice_jogo_eliminar, str_jogo_eliminar = pesquisar_jogo(lista_jogos)

    if indice_jogo_eliminar == -1:
        print("Nenhum jogo seleccionado.")
        return
```

```
    print('\n--- APAGAR Jogo ---')
    confirmar_eliminar = input("\nTem a certeza que quer APAGAR este jogo? (S/N):
").strip().lower()

    if confirmar_eliminar == 's':
        lista_jogos.pop(indice_jogo_eliminar)
        print("Jogo apagado com sucesso!")
        input('\nPrimir ENTER para continuar.')
    else:
        print("Operação cancelada.")

    input('\nPrimir ENTER para continuar.')
    return lista_jogos

''' --- Funcao CONSULTAR TODOS OS JOGOS ---
O Algoritmo da Edição:
1 - Apresentar 1 tabela por jornada
2 - Carregar ENTER para voltar ao menu
'''
def consultar_todos_jogos(lista_jogos):
    validacoes.limpar_ecra()
    print('--- Consultar Todos os Jogos ---\n')

    print('--- NOTA: jogos por realizar tem "golos" = -1')
    input("\nPressione ENTER para continuar.")
    if len(lista_jogos) == 0:
        print("Nenhum jogo registado.")
        input("\nPressione ENTER para continuar.")
        return

    # Agrupar os jogos por jornada
    jogos_por_jornada = {} # Vamos criar um dicionário para agrupar os jogos por
jornada
    for jogo in lista_jogos: # cada jogo é uma lista [Jornada, Data, Casa, G_Casa,
G_Fora, Fora]
        jornada = jogo[0]   # extrair a jornada do jogo
        if jornada not in jogos_por_jornada:    # isto vai criar a chave (Jornada) se
não existir
            jogos_por_jornada[jornada] = []     # cria uma lista vazia para essa
chave (jornada)
        jogos_por_jornada[jornada].append(jogo) # adiciona o jogo à lista da jornada
correspondente

    # Apresentar os jogos agrupados DE CADA jornada (chave do dicionário)
    validacoes.limpar_ecra()

    print('--- Consultar Todos os Jogos ---\n')
    print('--- NOTA: jogos por realizar tem "golos" = -1\n')
```

```
    for jornada, jogos in jogos_por_jornada.items():
        print(f"\n--- Jornada {jornada} ---")
        for jogo in jogos:
            print(f"  {jogo[1]} - {jogo[2]} vs {jogo[5]} ({jogo[3]}-{jogo[4]})")
    input("\nPressione ENTER para continuar.")
    return

'''--- MENU CONSULTAR
'''
def menu_consultar(lista_jogos):
    while True:
        validacoes.limpar_ecra()

        print("\n" + "="*30)
        print("      LIGA DE FUTEBOL - Menu de Consulta    ")
        print("="*30)
        print("1. Consultar todos os jogos")
        print("2. Consultar jogo especifico")
        print("0. Voltar ao menu principal")

        try:
            opcao = int(input("Escolha uma opção: "))

            if opcao == 1:
                consultar_todos_jogos(lista_jogos)
            elif opcao == 2:
                pesquisar_jogo(lista_jogos)
            elif opcao == 0:
                break
            else:
                print("Opção errada!")

        except:
            print("Opção inválida!")
            input()

'''--- MENU ElIMINAR
'''
def menu_eliminar(lista_jogos):
```

Ficheiro: estatisticas.py

```
# estatistica.py

'''--- Função MENU ESTATISTICAS ---
1. - Apresentar um menu com opções de estatísticas
2. - Permitir ao utilizador escolher uma opção e ver o resultado
3. - Voltar ao menu principal quando o utilizador escolher sair
'''

import os
import validacoes

def menu_estatisticas(lista_jogos):

    while True:
        validacoes.limpar_ecra()
        print("\n" + "="*30)
        print("      LIGA DE FUTEBOL - Menu de Estatisticas    ")
        print("="*30)
        print("1. Jogos com mais golos")
        print("2. Média de golos por jogo")
        print("3. Equipa com mais golos marcados")
        print("4. Equipa com mais golos sofridos")
        print("0. Voltar ao menu principal")

        try:
            opcao = int(input("Escolha uma opção: "))
        except:
            print("Opção inválida!")                  # se der erro na conversao para
inteiro
            input()
        else:
            if opcao == 1:
                jogos_mais_golos(lista_jogos)

            elif opcao == 2:
                media_golos_por_jogo(lista_jogos)

            elif opcao == 3:
                equipa_mais_golos(lista_jogos)

            elif opcao == 4:
                equipa_mais_golos_sofridos(lista_jogos)

            elif opcao == 0:
                break
            else:
```

```
                print("Opção errada!")

''' --- Função para CALCULAR TOTAL DE GOLOS DE UM JOGO ---
- vou usar na função de jogos com mais golos
- vou usar na função média de golos por jogo
Recebe uma lista de um jogo e devolve a soma dos golos.
'''
def calcular_total_golos(jogo):
    # jogo[3] são os golos da casa, jogo[4] são os golos de fora
    return jogo[3] + jogo[4]
# --- Fim Função CALCULAR TOTAL GOLOS JOGO ---

''' --- Função  JOGOS COM MAIS GOLOS ---
1. - Ordenar a lista de jogos com base no total de golos (golos_casa + golos_fora) em
ordem decrescente
2. - Selecionar os 10 primeiros jogos da lista ordenada

Regras:
- Validação: se lista vazia devolve lista vazia
- top_n coerente: se <= 0 devolve lista vazia; se > len(lista) devolve todos.
- Ordenação: por total_golos desc; como desempate mantém ordem original (estável).
- Retorno: lista de dicts (cópias dos jogos) com campo adicional 'total_golos' para
facilitar impressão.

'''
def jogos_mais_golos(lista_jogos, top_n_default=10):
    """
    Top jogos por total de golos, assumindo cada jogo no formato: [jornada, data,
equipa_casa, golos_casa, golos_fora, equipa_fora]
    Retorna uma lista de listas do mesmo formato com um campo adicional no final:
total_golos
    """
    # 1) Validação básica
    if lista_jogos == []:
        validacoes.limpar_ecra()
        print("Nenhum jogo registado.")
        input("\nPressione ENTER para continuar.")
        return []

    # 2) Calcular total de golos por jogo (defensivo)
    jogos_com_totais = []

    for jogo in lista_jogos:

        total = calcular_total_golos(jogo)  # usamos a função de contar total golos
por jogo

        novo = [jogo[0], jogo[1], jogo[2], jogo[3], jogo[4], jogo[5], total]
```

```
        jogos_com_totais.append(novo) # vou adicionando o jogo com total de golos à
lista

    # 4) Ordenar por total (descendente). sort/ sorted é estável para empates.
    jogos_ordenados = sorted(jogos_com_totais, key=lambda x: x[6], reverse=True) #
key=lambda x: x[6] – para cada elemento x da lista (aqui uma lista por jogo) a função
lambda retorna x[6] (o indice 6, total_golos) que é usado como chave de ordenação.

    # 5) Selecionar top_n (se top_n > len, devolve tudo)
    top_jogos = jogos_ordenados[:top_n_default]

    # 6) Mostrar resultados ao utilizador
    validacoes.limpar_ecra()
    print(f"\nTop {top_n_default} jogos com mais golos:\n")
    for i in top_jogos:
        print(f"Jornada {i[0]} | {i[1]} | {i[2]} {i[3]} - {i[4]} {i[5]}  (Total:
{i[6]})")
    input("\nPressione ENTER para continuar.")
# --- Fim Função JOGO com MAIS GOLOS  ---

''' --- Função EQUIPA com MAIS GOLOS MARCADOS ---
1. - Criamos um dicionário vazio.
2. - Percorremos a lista de jogos com um ciclo for.
3. - Para cada jogo, vamos buscar quem jogou e quantos golos marcou.
4. - Somamos esses golos ao total da equipa no dicionário.
5. - No fim, fazemos outro ciclo for apenas para descobrir quem tem o valor mais
alto.
'''
def equipa_mais_golos(lista_jogos):

    golos_por_equipa = {}       # dicionáro que vai guardar chave:valor
('equipa':golos marcados)

    # Validação se lista vazia
    if lista_jogos == []:
        validacoes.limpar_ecra()
        print("Nenhum jogo registado.")
        input("\nPressione ENTER para continuar.")
        return []

    for jogo in lista_jogos:

        # nomes equipas e golos
        nome_casa = jogo[2]
        golos_casa = jogo[3]

        nome_fora = jogo [5]
        golos_fora = jogo[4]
```

```
        # --- Somar golos da Equipa da Casa ---
        if nome_casa in golos_por_equipa:
            # Se já existe, então somamos ao valor de golos que já lá estava
            golos_por_equipa[nome_casa] = golos_por_equipa[nome_casa] + golos_casa #
basta indicar que vamos somar (???)
        else:
            # Se não existe, cria a entrada nova com o nome da equipa e os golos
deste jogo
            golos_por_equipa[nome_casa] = golos_casa

        # --- repetimos o racional agora para somar golos da Equipa de Fora ---
        if nome_fora in golos_por_equipa:
            golos_por_equipa[nome_fora] = golos_por_equipa[nome_fora] + golos_fora
        else:
            golos_por_equipa[nome_fora] = golos_fora

    # 3. Descobrir qual a equipa com mais golos (Algoritmo do Maior)
    equipas_mais_golos = []
    max_golos = -1

    # 4. Percorrer o dicionário (chave é o nome, valor são os golos)
    for equipa in golos_por_equipa:
        total_desta_equipa = golos_por_equipa[equipa]

        # CASO 1: Encontrámos um novo recorde!
        if total_desta_equipa > max_golos:
            max_golos = total_desta_equipa        # Atualizamos o recorde
            lista_vencedores = [equipa]           # IMPORTANTE: Criamos uma lista
NOVA (apagamos os anteriores), pois temos um novo record

        # CASO 2: É um empate com o recorde atual
        elif total_desta_equipa == max_golos:
            lista_vencedores.append(equipa)       # Aqui usamos APPEND (junta-se aos
lideres já encontrados)

    texto_equipas = ""
    for nome in lista_vencedores:
        if texto_equipas == "":
            texto_equipas = nome
        else:
            texto_equipas = texto_equipas + " e " + nome
    validacoes.limpar_ecra()
    print(f"\nA equipa com mais golos marcados é: {texto_equipas}")
    print(f"Total de golos: {max_golos}")

    # Pausa para o utilizador ler
    input("\nPressione ENTER para continuar.")
# --- Fim Função EQUIPA com MAIS GOLOS MARCADOS ---
```

```
''' --- Função EQUIPA com MAIS GOLOS SOFRIDOS (Versão Procedimento) ---
1. - Contar os golos sofridos
2. - Encontrar a(s) equipa(s) com mais golos sofridos (pior defesa)
3. - Imprimir diretamente o resultado
'''
def equipa_mais_golos_sofridos(lista_jogos):

    if lista_jogos == []:
        validacoes.limpar_ecra()
        print("Nenhum jogo registado.")
        input("\nPressione ENTER para continuar.")
        return # Sai do procedimento

    golos_sofridos_por_equipa = {}

    # 2. Somar golos SOFRIDOS
    for jogo in lista_jogos:

        nome_casa = jogo[2]
        golos_casa = jogo[3] # Golos que a equipa de Casa MARCOU

        nome_fora = jogo[5]
        golos_fora = jogo[4] # Golos que a equipa de Fora MARCOU

        # Equipa da CASA sofre os golos da equipa de FORA ---
        if nome_casa in golos_sofridos_por_equipa:
            golos_sofridos_por_equipa[nome_casa] =
golos_sofridos_por_equipa[nome_casa] + golos_fora
        else:
            golos_sofridos_por_equipa[nome_casa] = golos_fora

        # Equipa de FORA sofre os golos da equipa de CASA ---
        if nome_fora in golos_sofridos_por_equipa:
            golos_sofridos_por_equipa[nome_fora] =
golos_sofridos_por_equipa[nome_fora] + golos_casa
        else:
            golos_sofridos_por_equipa[nome_fora] = golos_casa

    # Descobrir qual a equipa com mais golos sofridos (Pior defesa)
    lista_piores_defesas = []
    max_golos_sofridos = -1

    for equipa in golos_sofridos_por_equipa:
        total_sofridos = golos_sofridos_por_equipa[equipa]

        if total_sofridos > max_golos_sofridos:
            max_golos_sofridos = total_sofridos    # Novo recorde negativo
            lista_piores_defesas = [equipa]        # Reinicia a lista
```

```
        elif total_sofridos == max_golos_sofridos:
            lista_piores_defesas.append(equipa)    # Empate

    # 4. Formatar texto e mostrar resultado
    texto_equipas = ""
    for nome in lista_piores_defesas:
        if texto_equipas == "":
            texto_equipas = nome
        else:
            texto_equipas = texto_equipas + " e " + nome

    validacoes.limpar_ecra()
    print(f"\nA equipa com mais golos SOFRIDOS é: {texto_equipas}")
    print(f"Total de golos sofridos: {max_golos_sofridos}")

    input("\nPressione ENTER para continuar.")
# --- Fim Função EQUIPA com MAIS GOLOS SOFRIDOS ---


''' --- Função MEDIA DE GOLOS POR JOGO ---
1. - Iterarar todos os jogos da lista de jogos
2. - Criar um dicionario
Somar todos os golos marcados em todos os jogos
2. - Dividir o total de golos pelo número total de jogos
'''
def media_golos_por_jogo(lista_jogos):

    if len(lista_jogos) == 0:
        validacoes.limpar_ecra()
        print("Nenhum jogo registado.")
        input("\nPressione ENTER para continuar.")
        return -1

    total_golos_campeonato = 0
    total_jogos = len(lista_jogos)

    for jogo in lista_jogos:
        # AQUI: Usamos a função auxiliar que criámos!
        golos_deste_jogo = calcular_total_golos(jogo)

        # Adicionamos ao acumulador
        total_golos_campeonato += golos_deste_jogo

    media = total_golos_campeonato / total_jogos
    validacoes.limpar_ecra()
    print(f'A média foi de, {media:.2f} golos/jogo\n')
    input("\nPressione ENTER para continuar.")
# --- Fim Função MEDIA DE GOLOS POR JOGO ---
```

```
'''--- Funcao TABELA CLASSIFICACAO ---
1 - Calcular os pontos de cada equipa (Vitoria = 3, Empate = 1, Derrota = 0)
2 - Ordenar a tabela por pontos
3 - Apresentar a tabela de classificação
4 - Carregar ENTER para voltar ao menu
'''
def tabela_classificacao(lista_jogos):

    # 1. Validação se lista vazia
    if lista_jogos == []:
        print("Nenhum jogo registado.")
        input("\nPressione ENTER para continuar.")
        return

    # Dicionário, onde a chave é o nome da equipa, o valor é uma lista: [Pontos,
Jogos]
    info_equipa = {}

    for jogo in lista_jogos:
        # Extrair dados: [Jornada, Data, equipa Casa, GC, GF, equipa Fora]
        casa = jogo[2].lower()
        gc = jogo[3]
        fora = jogo[5].lower()
        gf = jogo[4]

        # Se a equipa não existe no dicionário, cria-se a entrada a zeros
        if casa not in info_equipa:
            info_equipa[casa] = [0, 0] # isto cria a chave:valro --> equipa:[Pontos,
Jogos]
        if fora not in info_equipa:
            info_equipa[fora] = [0, 0]

        # --- Atualizar Jogos Realizados (índice 1) ---
        info_equipa[casa][1] += 1
        info_equipa[fora][1] += 1

        # --- Calcular Pontos (índice 0) ---
        if gc > gf:                     # Vitória da Casa
            info_equipa[casa][0] += 3   # Casa ganha 3 pontos

        elif gf > gc:                   # Vitória de Fora
            info_equipa[fora][0] += 3   # Fora ganha 3 pontos

        else:             # Empate
            info_equipa[casa][0] += 1   # 1 Ponto para casa
            info_equipa[fora][0] += 1   # 1 Ponto para fora

    # Agora converter Dicionário para Lista (para poder ordenar)
    # Formato final da lista: [Nome, Pontos, Jogos]
```

```
    tabela_final = []

    for nome_equipa in info_equipa:
        dados = info_equipa[nome_equipa]    # vou receber a lista [Pontos, Jogos] de
cada nome_equipa do dicionário
        pontos = dados[0]                   # a cada lista vou ao indice 0, buscar os
Pontos
        jogos = dados[1]                    # Vou ao indice 1 buscar os Jogos

        # Criar a linha e adicionar à tabela (lista de listas)
        linha = [nome_equipa, pontos, jogos]
        tabela_final.append(linha)

    # Ordenar a Tabela

    tabela_ordenada = sorted(tabela_final, key=lambda x: x[1], reverse=True)
    # key=lambda x: x[1] -> Ordena pelos Pontos (que está no índice 1 da nossa lista
'linha').
    # reverse=True -> Do maior para o menor
    # Já temos a lista de listas ordenada do primeiro para o último
    # Apresentar Tabela bonitinha

    validacoes.limpar_ecra()

    print(f"\n{'POS':<4} {'EQUIPA':<30} {'PTS':<5} {'J':<5}")   # o cabeçalho da
tabela
    print("="*45)

    posicao = 1     # contador para marcar a posição na tabela, de cada equipa
    for equipa in tabela_ordenada:
        # equipa = [Nome, Pontos, Jogos]
        nome = equipa[0]
        pts = equipa[1]
        j = equipa[2]

        print(f"{posicao:<4} {nome:<30} {pts:<5} {j:<5}")
        posicao += 1

    print("="*45)
    input("\nPressione ENTER para voltar.")

# --- Fim Função TABELA CLASSIFICACAO ---
```

Conclusões

O desenvolvimento deste projeto permitiu consolidar os conhecimentos adquiridos na unidade curricular de Fundamentos de Programação, em particular na utilização da linguagem Python. O objetivo principal de criar um sistema de gestão para uma liga de futebol foi atingido, integrando funcionalidades CRUD (Create, Read, Update and Delete) e adicionalmente o consumo de APIs externas.

Principais Desafios: A maior dificuldade encontrada residiu na manipulação das estruturas de dados necessárias para gerar a Tabela de Classificação com funções nativas de python (sem recorrer a biblioteca pandas, por exemplo). A lógica de agrupar equipas, somar pontos de jogos em casa e fora, e ordenar por múltiplos critérios exigiu um estudo aprofundado sobre Dicionários e funções ***Lambda***. A integração da biblioteca **requests** também apresentou desafios iniciais na autenticação e tratamento de erros de conexão, mas neste caso, o recurso a ferramentas de AI foi vital para a concretização do objetivo.

Melhorias Futuras: Como trabalho futuro, o sistema poderia beneficiar da implementação de uma interface gráfica completa (GUI) para todas as funcionalidades, abandonando a consola. Além disso, seria interessante aprofundar a estatística, incluindo dados sobre jogadores (melhores marcadores) e não apenas equipas.

Bibliográficas e Referências

- **Python Software Foundation.** (2024). *Python 3.12 Documentation*. Disponível em: https://docs.python.org/3/
- **Football-Data.org.** (2024). *API Documentation*. Disponível em: https://www.football-data.org/documentation/quickstart
- **Requests: HTTP for Humans.** (s.d.). *Documentation*. Disponível em: https://requests.readthedocs.io/
- Apontamentos e material de apoio da unidade curricular de Fundamentos de Programação.