UNIVERSIDADE DE VASSOURAS – CAMPUS MARICÁ
ENGENHARIA DE SOFTWARE

HUGO LELLY DE LIMA MARINHO

ESTRUTURA DE DADOS

P2 - ENUNCIADO E REGRAS

MARICÁ, RJ
2023

UNIVERSIDADE DE VASSOURAS

INFORMAÇÕES:

CURSO:
ENGENHARIA DE SOFTWARE

DISCIPLINA:
ESTRUTURA DE DADOS

PROFESSOR:
MÁRCIO GARRIDO

TÍTULO:
P2 - ENUNCIADO E REGRAS

TURMA:
2022.1 – TURMA A

MATRÍCULA:
202211182

ALUNO:
HUGO LELY DE LIMA MARINHO

"A educação tem raízes amargas, mas os seus frutos são doces."
Aristóteles, filósofo grego.

UNIVERSIDADE DE VASSOURAS

SUMÁRIO

UNIVERSIDADE DE VASSOURAS

ENUNCIADO:

A Universidade de Vassouras do Campus 1 foi convidada pela Prefeitura de Maricá para promover uma solução tecnológica em um dos problemas sociais da cidade, o abandono de animais. Mesmo considerado crime (O abandono de animais é crime, previsto na Lei de Crimes Ambientais - Lei Federal n° 9.605 de 1998), e notório que o índice de abandono vem crescendo a cada ano.

Os alunos do curso de Engenharia de Software foram convocados para a reunião com a secretaria da cidade para entender a demanda solicitada e alguns pontos foram levantados.

A prefeitura precisa de um sistema que possa cadastrar todos os animais por tipo (canino, felino etc.) e para tanto, é uma premissa que seja possível inserir novos tipos dinamicamente.

Precisa ainda, que sejam classificados por idade aproximada, cor, porte e se possui alguma particularidade.

No mesmo sistema, deverá ter também um cadastro de pessoas interessadas na adoção, contendo os dados principais de contato e qual espécie teria o interesse de adotar.

Ao escolher a espécie, deve também informar se possui alguma preferência do animal.

Por fim, no final do mês a prefeitura emitirá um relatório de cruzamento de espécies disponíveis x possíveis candidatos, ou quando um candidato a adoção ligar, que o atendente possa pesquisar se há algum animal com as características informadas.

Os alunos anotaram atentamente a todas as observações, criaram o fluxograma do estudo de caso, e posteriormente o primeiro protótipo em Python, ainda que em modo texto, e sem requisitos gráficos.

A ideia foi apenas validar a proposta do programa junto ao solicitante.

UNIVERSIDADE DE VASSOURAS

REGRAS DE AVALIAÇÃO:

FLUXOGRAMA EM PDF NO REPOSITÓRIO (1 PONTO)

- Organização clara (0,5 Pontos)
- Funcional de acordo com o enunciado (0,5 Pontos)

ESTRUTURA DE DADOS DO ALGORITMO (4 PONTOS)

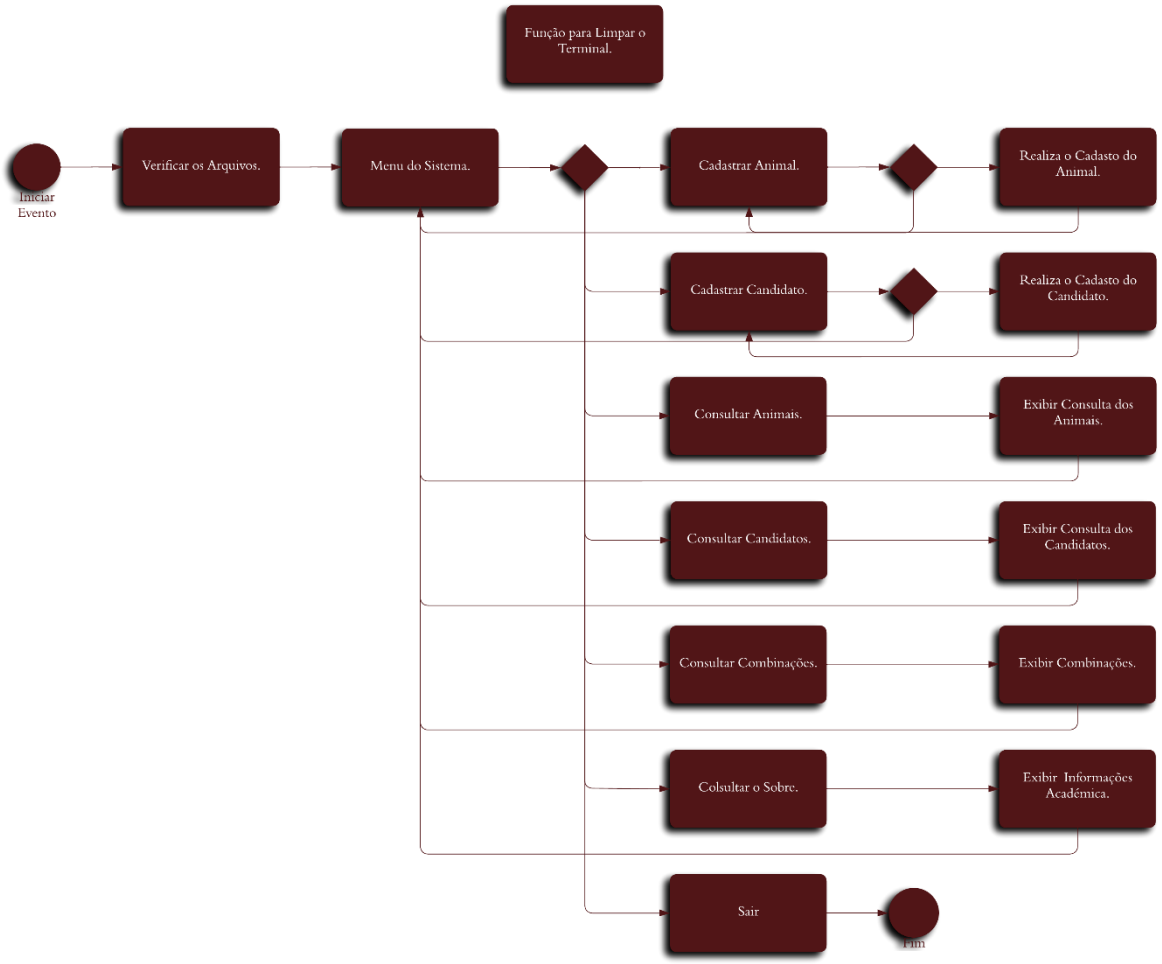
- Uso de ao menos 4 métodos de fila ou pilha (0,5 Ponto)
- Uso de ao menos 4 métodos recursivos (0,5 Pontos)
- Uso de pesquisa binária, lista encadeada (0,5 Pontos)
- O programa rodar com tratamento de erros - "entradas inválidas do usuário" (0,5 Pontos)
- O programa atender ao enunciado proposto (2 Pontos)

ORGANIZAÇÃO DO PROJETO (3 PONTOS)

- Hierarquia dos arquivos e organização das pastas - diretórios, nome de arquivos, classes, etc. (1 Ponto)
- Relação commit/dia com no mínimo 50 commits no final do projeto e no máximo 5 commits dia (1 Ponto)
- Organização do README do projeto contendo título do projeto, enunciado, participantes, nome do professor (linkando para meu GITHUB), disciplina e ao menos 3 imagens do fluxograma, código e o programa rodando. (1 Ponto)

FLUXOGRAMA:

Seguindo os critérios foi elaborado o seguinte fluxograma, com início, interações, exibições e fim.



CODIGO PYTHON:

O Sistema de Adoção foi dividido em 9 arquivos, para que cada função possa passar por manutenções futuras.

MENU.PY

Este arquivo é responsável por interagir com todos os outros códigos. Ele exibe um menu para o usuário e chama as funções correspondentes de acordo com a opção selecionada.

```
from time import sleep
from limparterminal import limpar_terminal
from cadastraranimal import cadastrar_animal
from cadastrarcandidato import cadastrar_candidato
from consultaranimais import consultar_animais
from consultarcandidatos import consultar_candidatos
from combinarinteresses import combinar_interesses
from exibirsobre import exibir_sobre
from verificararquivos import verificar_arquivos

verificar_arquivos()

# Função principal
def main():
    limpar_terminal()
    while True:
        print("\tBEM VINDO AO SISTEMA DE ADOÇÃO DE ANIMAIS")
        print("\n\tEscolha uma Opção:\n")
        print("\t1 - Cadastrar Animais")
        print("\t2 - Cadastrar Candidatos")
        print("\t3 - Consultar Animais Cadastrados")
        print("\t4 - Consultar Candidatos Cadastrados")
        print("\t5 - Realizar Combinações")
        print("\t6 - Sobre")
        print("\t7 - Sair")
        opcao = input("\n\tDigite a opção desejada: ")
        if opcao == "1":
            limpar_terminal()
            print("\tAbrindo o Cadastro de Animais")
            sleep(1.5)
            cadastrar_animal()
        elif opcao == "2":
            limpar_terminal()
            print("\tAbrindo o Cadastro de Candidatos")
            sleep(1.5)
            cadastrar_candidato()
        elif opcao == "3":
            limpar_terminal()
            print("\tAbrindo a Consulta de Animais")
```

```
            sleep(1.5)
            consultar_animais()
        elif opcao == "4":
            limpar_terminal()
            print("\tAbrindo a Consulta dos Candidatos")
            sleep(1.5)
            consultar_candidatos()
        elif opcao == "5":
            limpar_terminal()
            print("\tAbrindo as Combinações")
            sleep(1.5)
            combinar_interesses()
        elif opcao == "6":
            limpar_terminal()
            print("\tAbrindo o Sobre")
            sleep(1.5)
            exibir_sobre()
        elif opcao == "7":
            limpar_terminal()
            print("\tSaindo do Sistema\n\tAguarde...")
            sleep(1.5)
            limpar_terminal()
            print("\tOBRIADO POR UTILIZAR O\n\tSISTEMA DE ADOÇÃO DE ANIMAIS")
            print("\n\t🐶🐱🐮🐰")
            sleep(3)
            exit()
        else:
            limpar_terminal()
            print("\tOpção inválida!\n\tAgurade...")
            sleep(1.5)
            limpar_terminal()

if __name__ == "__main__":
    main()
```

VERIFICARARQUIVOS.PY

Este arquivo contém uma função que verifica a existência dos arquivos "animais.txt" e "candidatos.txt". Se os arquivos não existirem, eles serão criados.

```
import os

# Função para verificar se os arquivos txt existem. Se não existirem, serão
criados.
def verificar_arquivos():
    if not os.path.exists("animais.txt"):
        with open("animais.txt", "w", encoding="utf-8"):
            pass
```

```
    if not os.path.exists("candidatos.txt"):
        with open("candidatos.txt", "w", encoding="utf-8"):
            pass
```

LIMPARTERMINAL.PY

Este arquivo contém uma função que limpa o terminal. Isso é útil para manter o código limpo e organizado no terminal.

```
import os

# Função para limpar o terminal
def limpar_terminal():
    os.system("cls" if os.name == "nt" else "clear")
```

CADASTRARANIMAL.PY

Este arquivo contém uma função para cadastrar animais. Ele solicita ao usuário as informações necessárias sobre o animal e salva os dados no arquivo "animais.txt".

```
from time import sleep
from limparterminal import limpar_terminal

# Função para cadastrar animais
def cadastrar_animal():
    limpar_terminal()

    while True:
        print("\tCADASTRO DE ANIMAIS\n")
        print("\tDeseja Cadastrar Um Novo Animal?\n")

        opcao = input("\n\t1 - Sim\n\t2 - Não\n\n\tDigite a Opção Desejada:
")

        if opcao == "1":
            limpar_terminal()
            print("\tAbrindo Cadastro...")
            sleep(1.5)
            limpar_terminal()
            print("\tCADASTRAR NOVO ANIMAL")
            print("\n\tInforme os Dados do Animal:\n")
            sleep(1.5)
            especie = input("\tEspécie: ")
            raca = input("\tRaça: ")
            idade = input("\tIdade (meses): ")
            cor = input("\tCor: ")
            particularidades = input("\tParticularidades: ")
```

```
            sleep(1.5)

            with open("animais.txt", "a", encoding="utf-8") as arquivo:
                arquivo.write(f"{especie},{raca},{idade},{cor},{particularida
des}")

            print("\n\tCadastro Realizado com Sucesso!")
            sleep(1.5)
            limpar_terminal()

        elif opcao == "2":
            limpar_terminal()
            print("\tSaindo do Cadastro!\n\tAguarde...")
            sleep(1.5)
            limpar_terminal()
            return

        else:
            limpar_terminal()
            print("\tOpção Inválida!\n\tAguarde...")
            sleep(1.5)
            limpar_terminal()
```

CADASTRARCANDIDATO.PY

Este arquivo contém uma função para cadastrar candidatos. Ele solicita ao usuário as informações necessárias sobre o candidato e salva os dados no arquivo "candidatos.txt".

```
from time import sleep
from limparterminal import limpar_terminal

# Função para cadastrar candidatos
def cadastrar_candidato():
    limpar_terminal()

    while True:
        print("\tCADASTRO DE CANDIDATOS\n")
        print("\tDeseja Cadastrar Um Novo Candidato?\n")

        opcao = input("\n\t1 - Sim\n\t2 - Não\n\n\tDigite a Opção Desejada:
")

        if opcao == "1":
            limpar_terminal()
            print("\tAbrindo Cadastro...")
            sleep(1.5)
            limpar_terminal()
```

```
            print("\tCADASTRO DE CANDIDATOS")
            print("\n\tInforme os Dados do Candidato:\n")
            sleep(1.5)
            nome = input("\tNome: ")
            telefone = input("\tTelefone: ")
            email = input("\tE-mail: ")
            bairro = input("\tBairro: ")
            especie_interesse = input("\tEspécie de interesse: ")
            particularidades_interesse = input("\tParticularidades de
interesse: ")
            sleep(1.5)

            with open("candidatos.txt", "a", encoding="utf-8") as arquivo:
                arquivo.write(f"{nome},{telefone},{email},{bairro},{especie_i
nteresse},{particularidades_interesse}")

            print("\n\tCadastro Realizado com Sucesso!")
            sleep(1.5)
            limpar_terminal()

        elif opcao == "2":
            limpar_terminal()
            print("\n\tSaindo do Cadastro!\n\tAguarde...")
            sleep(1.5)
            limpar_terminal()
            return

        else:
            limpar_terminal()
            print("\n\tOpção Inválida!\n\tAguarde...")
            sleep(1.5)
            limpar_terminal()
```

CONSULTARANIMAIS.PY

Este arquivo contém uma classe “Nodo” e uma classe “ListaEncadeada”, que são usadas para realizar a consulta de animais cadastrados. Ele lê os dados do arquivo "animais.txt" e permite pesquisar por animais com base em determinados critérios.

```
from time import sleep
from limparterminal import limpar_terminal

class Nodo:
    def __init__(self, data):
        self.data = data
        self.next = None

class ListaEncadeada:
```

```
    def __init__(self):
        self.head = None

    def adicionar_elemento(self, data):
        novo_nodo = Nodo(data)
        if self.head is None:
            self.head = novo_nodo
        else:
            atual = self.head
            while atual.next:
                atual = atual.next
            atual.next = novo_nodo

    def pesquisar_elemento(self, campo_consulta):
        resultados = []
        atual = self.head
        while atual:
            # Lógica de pesquisa aqui (por exemplo, pesquisa binária)
            if campo_consulta.lower() in atual.data[0].lower() or
campo_consulta.lower() in atual.data[1].lower() or campo_consulta.lower() in
atual.data[2].lower() or campo_consulta.lower() in atual.data[3].lower() or
campo_consulta.lower() in atual.data[4].lower():
                resultados.append(atual.data)
            atual = atual.next
        return resultados

# Função para consultar animais cadastrados
def consultar_animais():
    limpar_terminal()
    print("\tCONSULTA DE ANIMAIS\n")

    campo_consulta = input("\tDigite a Consulta: ")
    campo_consulta = campo_consulta.lower()

    animais = ListaEncadeada()

    with open("animais.txt", "r", encoding="utf-8") as arquivo:
        for linha in arquivo:
            especie, raca, idade, cor, particularidades =
linha.strip().split(",")
            animais.adicionar_elemento((especie, raca, idade, cor,
particularidades))

    animais_encontrados = animais.pesquisar_elemento(campo_consulta)

    limpar_terminal()
    print("\tCONSULTA DE ANIMAIS\n")
    print(f"\t{len(animais_encontrados)} Animais Encontrados:")
    print("\t-----------------------------------\n")
```

```
    sleep(1.5)

    for animal in animais_encontrados:
        sleep(0.5)
        print(f"\tEspécie: {animal[0]}")
        print(f"\tRaça: {animal[1]}")
        print(f"\tIdade (Meses): {animal[2]}")
        print(f"\tCor: {animal[3]}")
        print(f"\tParticularidades: {animal[4]}")
        sleep(0.5)
        print("\t---------------------------------")

    sleep(1.5)
    input("\n\tPressione Enter para voltar ao menu principal.")
    limpar_terminal()
```

CONSULTARCANDIDATOS.PY

Este arquivo é semelhante ao arquivo "consultaranimais.py", mas é usado para consultar candidatos cadastrados com base em critérios específicos.

```
from time import sleep
from limparterminal import limpar_terminal

class Nodo:
    def __init__(self, data):
        self.data = data
        self.next = None

class ListaEncadeada:
    def __init__(self):
        self.head = None

    def adicionar_elemento(self, data):
        novo_nodo = Nodo(data)
        if self.head is None:
            self.head = novo_nodo
        else:
            atual = self.head
            while atual.next:
                atual = atual.next
            atual.next = novo_nodo

    def pesquisar_elemento(self, campo_consulta):
        resultados = []
        atual = self.head
        while atual:
            # Lógica de pesquisa aqui (por exemplo, pesquisa binária)
```

```
            if campo_consulta.lower() in atual.data[0].lower() or
campo_consulta.lower() in atual.data[1].lower() or campo_consulta.lower() in
atual.data[2].lower() or campo_consulta.lower() in atual.data[3].lower() or
campo_consulta.lower() in atual.data[4].lower() or campo_consulta.lower() in
atual.data[5].lower():
                resultados.append(atual.data)
            atual = atual.next
        return resultados

# Função para consultar candidatos cadastrados
def consultar_candidatos():
    limpar_terminal()
    print("\tCONSULTA DE CANDIDATOS\n")

    campo_consulta = input("\tDigite a Consulta: ")
    campo_consulta = campo_consulta.lower()

    candidatos = ListaEncadeada()

    with open("candidatos.txt", "r", encoding="utf-8") as arquivo:
        for linha in arquivo:
            nome, telefone, email, bairro, especie_interesse,
particularidades_interesse = linha.strip().split(",")
            candidatos.adicionar_elemento((nome, telefone, email, bairro,
especie_interesse, particularidades_interesse))

    candidatos_encontrados = candidatos.pesquisar_elemento(campo_consulta)

    limpar_terminal()
    print("\tCONSULTA DE CANDIDATOS\n")
    print(f"\t{len(candidatos_encontrados)} Candidatos Encontrados:")
    print("\t-----------------------------------\n")
    sleep(1.5)

    for candidato in candidatos_encontrados:
        sleep(0.5)
        print(f"\tNome: {candidato[0]}")
        print(f"\tTelefone: {candidato[1]}")
        print(f"\tE-mail: {candidato[2]}")
        print(f"\tBairro: {candidato[3]}")
        print(f"\tEspécie de Interesse: {candidato[4]}")
        print(f"\tParticularidades de Interesse: {candidato[5]}")
        sleep(0.5)
        print("\t-----------------------------------")

    sleep(1.5)
    input("\n\tPressione Enter para voltar ao menu principal.")
    limpar_terminal()
```

COMBINARINTERESSES.PY

Este arquivo contém uma função para combinar os interesses de candidatos e animais. Ele lê os dados dos arquivos "animais.txt" e "candidatos.txt" e exibe as combinações encontradas.

```
from time import sleep
from limparterminal import limpar_terminal

# Função para combinar interesses de candidatos e animais
def combinar_interesses():
    limpar_terminal()
    print("\tCOMBINAÇÕES\n")
    sleep(1.5)
    print("\tCombinações Encontradas")
    print("\t-----------------------\n")
    sleep(1.5)

    animais = []
    candidatos = []

    with open("animais.txt", "r", encoding="utf-8") as arquivo:
        for linha in arquivo:
            especie, raca, idade, cor, particularidades =
linha.strip().split(",")
            animais.append((especie, raca, idade, cor, particularidades))

    with open("candidatos.txt", "r", encoding="utf-8") as arquivo:
        for linha in arquivo:
            nome, telefone, email, bairro, especie_interesse,
particularidades_interesse = linha.strip().split(",")
            candidatos.append((nome, telefone, email, bairro,
especie_interesse, particularidades_interesse))

    animais_encontrados = []
    candidatos_encontrados = []

    for animal in animais:
        for candidato in candidatos:
            if animal[0].lower() == candidato[4].lower() and
animal[4].lower() == candidato[5].lower():
                animais_encontrados.append(animal)
                candidatos_encontrados.append(candidato)

    print(f"\n\t{len(animais_encontrados)} Animais Encontrados:\n")
    print("\t---------------------------------")

    for animal in animais_encontrados:
```

```
        sleep(0.5)
        print(f"\tEspécie: {animal[0]}")
        print(f"\tRaça: {animal[1]}")
        print(f"\tIdade: {animal[2]}")
        print(f"\tCor: {animal[3]}")
        print(f"\tParticularidades: {animal[4]}")
        sleep(0.5)
        print("\t----------------------------------")

    sleep(1.5)
    print(f"\n\t{len(candidatos_encontrados)} Candidatos Encontrados:\n")
    print("\t----------------------------------")

    for candidato in candidatos_encontrados:
        sleep(0.5)
        print(f"\tNome: {candidato[0]}")
        print(f"\tTelefone: {candidato[1]}")
        print(f"\tE-mail: {candidato[2]}")
        print(f"\tBairro: {candidato[3]}")
        print(f"\tEspécie de Interesse: {candidato[4]}")
        print(f"\tParticularidades de Interesse: {candidato[5]}")
        sleep(0.5)
        print("\t----------------------------------")

    sleep(1.5)
    input("\n\tPressione Enter para voltar ao menu principal.")
    limpar_terminal()
```

EXIBIRSOBRE.PY

Na função sobre, e printado no terminal os dados do trabalho acadêmico.

```
from time import sleep
from limparterminal import limpar_terminal

# Função para exibir informações sobre o sistema
def exibir_sobre():
    limpar_terminal()
    print("\tSISTEMA DE ADOÇÃO DE ANIMAIS")
    print("\n\tDesenvolvido por:\n\tHUGO LELY DE LIMA MARINHO\n")
    sleep(5)
    limpar_terminal()
    print("\tUNIVERSIDADE DE VASSOURAS")
    print("\t  CAMPUS MARICÁ")
    sleep(1.5)
    print("\tCURSO:")
    print("\t  ENGENHARIA DE SOFTWARE")
    sleep(1.5)
```

```
    print("\tDISCIPLINA:")
    print("\t  ESTRUTURA DE DADOS")
    sleep(1.5)
    print("\tPROFESSOR:")
    print("\t  MÁRCIO GARRIDO")
    sleep(1.5)
    print("\tTÍTULO:")
    print("\t  P2 - SISTEMA DE ADOÇÃO DE ANIMAIS")
    sleep(1.5)
    print("\tTURMA:")
    print("\t  2022.1 – TURMA A")
    sleep(1.5)
    print("\tMATRÍCULA:")
    print("\t  202211182")
    sleep(1.5)
    print("\tALUNO:")
    print("\t  HUGO LELY DE LIMA MARINHO")
    sleep(10)
    print("\n\tSaindo do Sobre!\n\tAguarde...")
    sleep(1.5)
    limpar_terminal()
```