Documentação do Código “Vida Pet”

Este documento descreve as principais partes e funções do código `Vida Pet`, que gerencia pets, eventos, metas e um “Tinder de Pets” usando arquivos de texto.

1. Visão Geral

- **Objetivo**: permitir que tutores cadastrem pets, registrem eventos de saúde, definam metas de bem-estar, recebam sugestões de cuidados e façam “match” entre pets da mesma espécie.

- **Persistência**: todos os dados são guardados em arquivos `.txt` usando o caractere `|` como separador de campos.

- **Público-alvo**: alunos de primeiro período de Ciência da Computação, com código didático, claro e sem bibliotecas.

2. Arquivos de Dados

Arquivo	Campos (separados por `|`)
`pets.txt`	`nome|especie|raca|nascimento|peso`
`eventos.txt`	`nome_pet|tipo_evento|data_evento|observacoes`
`metas.txt`	`nome_pet|descricao_meta|tipo_meta|frequencia|vezes_cumpridas`
`caixapostal_<pet>.txt`	linhas com pedidos de encontro de outros pets

3. Fluxo de Inicialização

1. **`inicializar_arquivos()`**

    - Garante existência de `pets.txt`, `eventos.txt` e `metas.txt`.

    - Se não existir, cria arquivo vazio.

2. **`main()`**

Limpa tela e apresenta o menu principal:

1 – Pets
2 – Eventos
3 – Sugestões
4 – Metas
5 – Tinder de Pets
0 – Sair

    - Redireciona para o submenu correspondente ou encerra.

4. Biblioteca de Funções de I/O

- **`ler_registros(caminho, campos_esperados)`**
  Lê um `.txt`, faz `split("|")`, ignora linhas vazias ou malformadas e retorna lista de listas de strings.

- **`salvar_registros(caminho, lista_de_listas)`**
  Recebe lista de listas e regrava o arquivo, unindo cada sublista com `"|"` e adicionando `"\n"`.

5. Submenus

5.1 Pets (CRUD)

- **`adicionar_pet()`**

    - Pede: nome, espécie, raça, nascimento (DD/MM/AAAA) e peso (kg).

    - Valida data e peso.

    - Anexa linha em `pets.txt`.

- **`listar_pets()`**
    - Lê todos os pets e exibe numerados.
    - Retorna a lista de registros.
- **`editar_pet()`**
    - Mostra lista, pede número, escolhe campo, solicita novo valor e valida.
    - Chama `salvar_registros` para atualizar `pets.txt`.
- **`excluir_pet()`**
    - Mostra lista, pede número e remove o registro da lista em memória.
    - Regrava o arquivo.

5.2 Eventos

- **`adicionar_evento()`**
    - Verifica existência do pet em `pets.txt`.
    - Pede tipo, data (valida) e observações.
    - Anexa linha em `eventos.txt`.
- **`listar_eventos()`**
    - Lê `eventos.txt`, pede nome do pet e filtra as linhas.
    - Exibe numeradas ou mensagem de “nenhum evento”.

5.3 Sugestões

- **`sugestoes()`**
    - Lê `pets.txt`, busca registro do pet.

- Calcula idade em anos com `datetime.strptime` + diferença de datas.
- Exibe recomendações de brinquedos e exercícios baseadas em espécie e idade.

5.4 Metas (CRUD)

- **`adicionar_meta()`**
  - Verifica existência do pet.
  - Pede descrição, tipo (1-Única/2-Recorrente) e frequência.
  - Anexa linha em `metas.txt` com contador `0`.
- **`listar_metas()`**
  - Lê `metas.txt`, filtra por pet e exibe numeradas com contadores.
- **`marcar_meta()`**
  - Lê `metas.txt`, coleta índices das metas do pet.
  - Pede número, incrementa o contador e regrava arquivo.

5.5 Tinder de Pets

- **`tinder_pets()`**
  - Verifica cadastro do pet.
  - Pede descrição e sexo.
  - Armazena em dicionário `cadastros_tinder[nome_pet] = [espécie, sexo, descrição]`.
  - **Opção 1**: lista candidatos (mesma espécie e sexo oposto), pede número e grava pedido em `caixapostal_<outro>.txt`.
  - **Opção 2**: lê `caixapostal_<nome_pet>.txt`, exibe pedidos, permite aceitar um e imprime confirmação.

6. Observações de Implementação

- Tratamos `FileNotFoundError` sempre que lemos um arquivo.
- Validamos formatos de data (`datetime.strptime`) e numéricos (`float()`).
- Menus em `while True` com opção “0” para voltar.