Descrição do Problema:

Camila é tutora de dois cachorros e um gato. Ela ama seus pets, mas tem dificuldade para lembrar datas de vacinação, banhos, consultas no veterinário, controle de vermífugos e alimentação especial. Pensando nisso, vamos criar o sistema **"Vida Pet"**, que ajudará Camila (e outros tutores) a organizar os cuidados com seus animais de estimação, mantendo a saúde e a rotina sempre atualizados.

Requisitos funcionais:

1. **CRUD de Animais de Estimação:**
   Camila poderá adicionar, visualizar, editar e excluir registros dos seus pets, com essas informações: nome, espécie, raça, data de nascimento e peso.
2. **Cadastro de Cuidados e Eventos:**
   O sistema permitirá o registro de eventos importantes, sendo eles: vacinações, consultas veterinárias e aplicação de remédios. Para cada evento deve ser registrado datas, o nome do pet e observações.
3. **Armazenamento de Dados:**
   Todos os registros serão salvos em um arquivo .csv ou .txt, garantindo que o histórico dos cuidados dos pets esteja sempre acessível.
4. **Metas de Saúde e Bem-Estar:**
   Camila poderá definir metas como "levar ao veterinário a cada 6 meses" ou "fazer 3 passeios por semana" e acompanhar se está cumprindo ou não.
5. **Sugestões Personalizadas de Cuidados:**
   Com base na espécie e idade do pet, o sistema poderá sugerir cuidados extras como brinquedos ideais, alimentos recomendados ou exercícios adequados.
6. **Funcionalidade Extra:**
   Implementem uma funcionalidade extra. Sejam criativos.

Requisitos não funcionais:

1. Deve ser feito em Python sem o uso de bibliotecas adicionais.

Utilizar a linha de comando para entrada e saída (interação pelo terminal);

b. Exceções de bibliotecas:

os -> os.system("clear") ou "cls".

2. Os dados devem ser salvos em um arquivo no formato .csv ou .txt;

a. O trabalho deve ser feito em grupo.

b. Trabalhos que não forem feitos em grupo perderão 50% da nota.

3. O código deve estar organizado, portanto, deve conter:

a. Funções para dividir o código de forma lógica e evitar repetições;

b. Tratamento de exceções, para garantir que seu código esteja pronto para tratar casos inesperados.

c. Legibilidade do código, incluindo nomeação de variáveis e funções.

4. Deve ser feito um manual do usuário, explicando como utilizar a ferramenta e restrições gerais que a aplicação tenha.

a. Fiquem à vontade para escolher como será feito esse manual. Pode ser um pdf, site, vídeo, carta…

b. O manual é o readme que pode estar disponível no github do projeto (caso o grupo utilize o github, opcional, não será visto em sala de aula).

5. Não será aceito entregas atrasadas.

6. Apresentação:

a. A equipe deve apresentar o projeto feito para os professores.

b. Todos envolvidos da equipe devem explicar alguma parte, e perguntas direcionadas serão feitas durante a apresentação.

7. A entrega será em uma atividade do classroom

a. O que deve ser entregue:

Código da aplicação.

Manual do usuário.

Critérios de avaliação:

● Apresentação (50 pontos - nota individual):

- ○ Participação durante a apresentação do projeto;
- ○ Perguntas durante a apresentação.

● Código (50 pontos - nota por grupo):

- ○ Legibilidade e Organização do código;
- ○ Tratamento de erros;
- ○ Utilização de Arquivos;
- ○ Apresentação da ferramenta e manual do usuário;
- ○ Funcionalidade extra.