VetCare — Sistema de Clínica Veterinária

Aluno: Vanessa Martins

1. Aplicação dos Princípios SOLID no Projeto

O SOLID representa os cinco princípios fundamentais da programação orientada a objetos e design de código, propostos para facilitar a manutenção e a escalabilidade de softwares. São eles: *Single Responsibility Principle* (Princípio da Responsabilidade Única), *Open/Closed Principle* (Aberto/Fechado), *Liskov Substitution Principle* (Substituição de Liskov), *Interface Segregation Principle* (Segregação de Interface) e *Dependency Inversion Principle* (Inversão de Dependência).

Aplicação no Sistema VetCare

No desenvolvimento do VetCare, o princípio **SRP (Responsabilidade Única)** foi está na modularização do sistema. Cada classe, localizada em seu respectivo arquivo na pasta models (ex: tutor.py, animal.py), é responsável exclusivamente pela gestão de suas próprias regras de negócio e persistência de dados. Isso evita a criação de classes que centralizam muitas responsabilidades.

E o princípio **LSP (Substituição de Liskov)** foi aplicado na hierarquia de herança. As classes Tutor e Veterinario herdam da classe base Pessoa. Isso garante que ambas as subclasses possam usufruir dos atributos comuns (nome, telefone) e comportamentos da superclasse, mantendo a consistência lógica do objeto "Pessoa" dentro do sistema.

2. Arquitetura de Software: Padrão MVC

Definição do Padrão

O MVC (*Model-View-Controller*) é um padrão de arquitetura de software que separa a aplicação em três componentes lógicos principais: o **Model** (Modelo), responsável pelos dados e regras de negócio; a **View** (Visão),

responsável pela apresentação e interação com o usuário; e o **Controller** (Controlador), que intermedia a comunicação entre os dois.

Adaptação e Estrutura no Projeto VetCare

A estrutura foi organizada inspirada nos conceitos do MVC para garantir a organização e a separabilidade do código:

- **Model (Modelo):** Representado pela pasta models, contendo as classes (Tutor, Animal, Consulta, etc.) e o arquivo conexao.py. Esta camada é a única que se comunica diretamente com o Banco de Dados MySQL, abstraindo a complexidade do SQL do restante do sistema.
- **View (Visão):** Representada pelas funções de entrada e saída no arquivo main.py (os comandos print para exibir menus e input para capturar dados). É a interface visual com a qual o usuário interage.
- **Controller (Controlador):** Também situado no main.py, atua nas funções lógicas (como cadastrar_tutor()). Estas funções capturam os dados da *View* (o que o usuário digitou), instanciam os objetos do *Model* e acionam os métodos de persistência (.salvar()), coordenando o fluxo da informação.

3. Convenções de Escrita e Estilo de Código

Para garantir a legibilidade e a padronização do código fonte, o projeto seguiu as diretrizes do PEP 8 (Python Enhancement Proposal 8), que é o guia de estilo oficial da linguagem Python. As convenções de nomenclatura foram aplicadas da seguinte forma:

- PascalCase (ou CapitalCamelCase): Utilizado para a nomeação de Classes. Nesta convenção, a primeira letra de cada palavra é maiúscula e não há separadores.

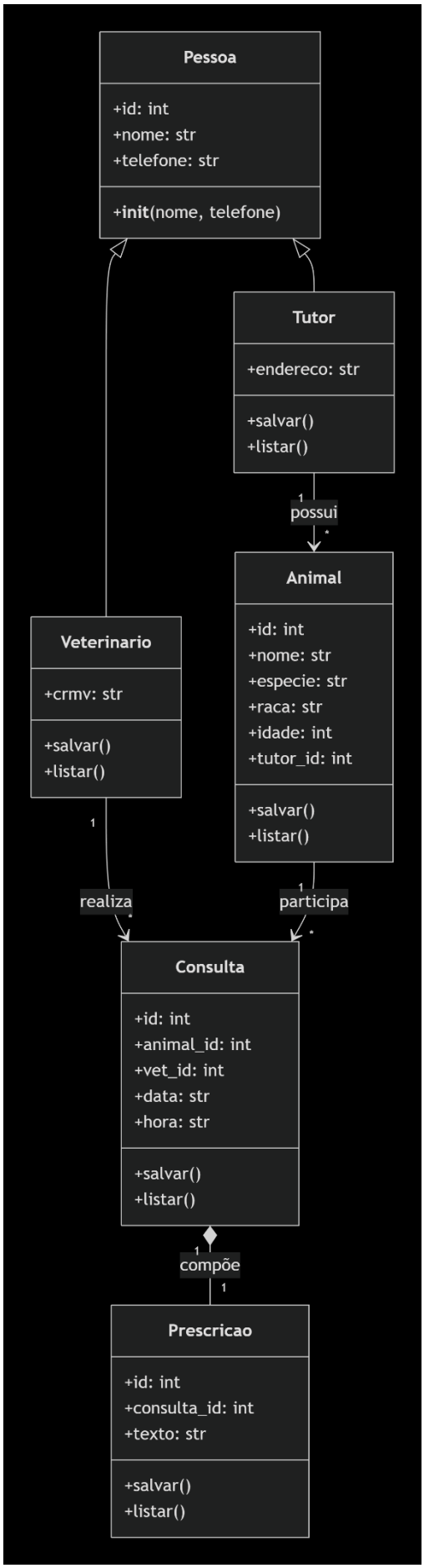
Exemplos no projeto: Veterinario, Tutor, Animal, Consulta.

- snake_case: Utilizado para a nomeação de Variáveis, Métodos e Funções. Nesta convenção, todas as letras são minúsculas e as palavras são separadas por um underscore (sublinhado).

Exemplos no projeto: listar_animais(), id_tutor, data_consulta, criar_conexao().

Justificativa

A escolha por essas convenções, em vez do camelCase, que é comum em Java ou JavaScript, deve-se pela função nativa dessas normas no Python. O uso correto de snake_case facilita a leitura visual de nomes longos de métodos, enquanto o PascalCase permite identificar rapidamente as entidades (Classes) do sistema, favorecendo a manutenção e o entendimento da estrutura do software por outros desenvolvedores.


Pessoa
+id: int
+nome: str
+telefone: str
+init(nome, telefone)
Tutor
+endereco: str
+salvar()
+listar()
1
possui
*
Animal
+id: int
+nome: str
+especie: str
+raca: str
+idade: int
+tutor_id: int
+salvar()
+listar()
Veterinario
+crmv: str
+salvar()
+listar()
1
realiza
*
1
participa
*
Consulta
+id: int
+animal_id: int
+vet_id: int
+data: str
+hora: str
+salvar()
+listar()
1
compõe
1
Prescricao
+id: int
+consulta_id: int
+texto: str
+salvar()
+listar()