Aluno: João Victor de Oliveira Melo

Professora: Lidiane Visintin

Desenvolvimento de Sistemas Orientados a Objetos

24 de abril de 2025

Relatório Explicativo - Sistema de Gestão para PetShop

Descrição do Cenário

O projeto consiste no desenvolvimento de um **Sistema de Gestão para PetShop**, cujo objetivo é facilitar o controle de clientes, funcionários e serviços oferecidos, como **banho**, **tosa** e **consultas veterinárias**.

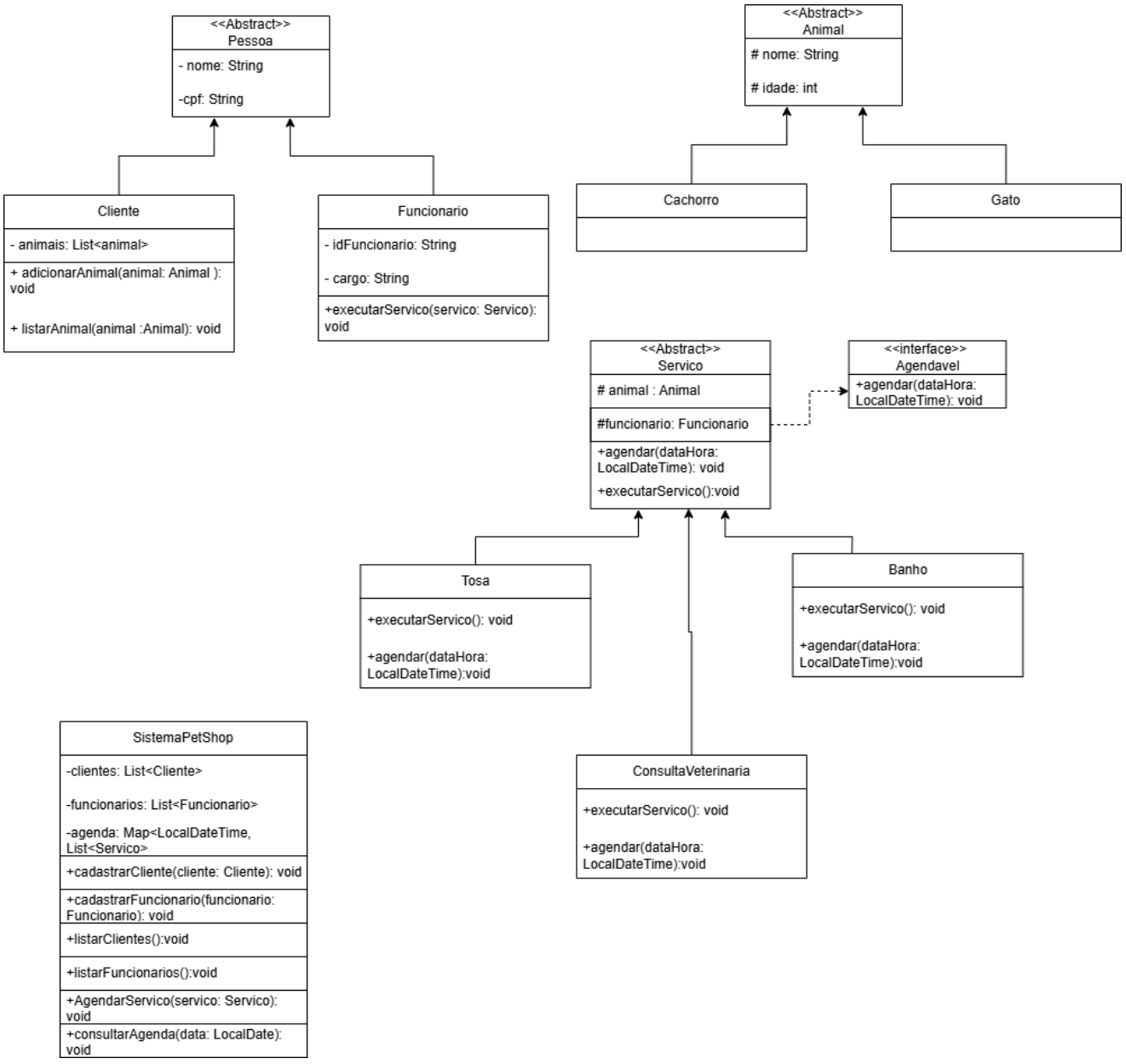
A escolha deste tema se deu por uma motivação pessoal: minha mãe é proprietária de um PetShop e frequentemente relata dificuldades em organizar e agendar os atendimentos de forma eficiente. A partir dessa necessidade real, surgiu a ideia de criar este sistema, permitindo aplicar na prática os conceitos aprendidos ao longo da disciplina de Programação Orientada a Objetos.

O sistema foi desenvolvido com foco na otimização do tempo, organização das informações e facilidade de uso, possibilitando:

- Cadastro de clientes com seus respectivos animais
- Cadastro de funcionários
- Agendamento de serviços com data e hora
- Consulta da agenda diária de atendimentos
- Visualização dos dados cadastrados

O sistema é baseado em uma estrutura orientada a objetos e foi implementado utilizando a linguagem Java.

Diagrama de Classes:



Justificativas da Modelagem

- **Herança (Pessoa, Animal e Servico):**
  Utilizamos herança para criar subclasses específicas como **Cliente**, **Funcionario**, **Cachorro**, **Gato**, **Banho**, **Tosa** e **ConsultaVeterinaria**, derivadas de classes abstratas (**Pessoa**, **Animal** e **Servico**). Essa herança permite o reaproveitamento de atributos comuns, como nome, CPF, idade, raça etc., evitando a duplicação de código e facilitando a manutenção.

- **Composição (Cliente → Animal):**
  Um cliente pode possuir vários animais. Essa relação foi representada usando uma **List<Animal>** dentro da classe **Cliente**, o que reflete de forma fiel a estrutura real de um PetShop.

- **Map para Agenda:**
  A agenda de serviços foi implementada utilizando **Map<LocalDateTime, List<Servico>>**, o que permite armazenar múltiplos serviços em um mesmo horário. Essa escolha garante eficiência na busca por agendamentos em horários específicos e permite fácil expansão da lógica da agenda.

- **Abstração e Polimorfismo:**
  A classe **Servico** é abstrata e define o método **executarServico()**, que é sobrescrito por suas subclasses (**Banho**, **Tosa**, **ConsultaVeterinaria**). Com isso, o sistema pode lidar com diferentes tipos de serviços de maneira uniforme e extensível, sem depender de verificações específicas para cada tipo.

- **Interface (Agendavel):**
  A interface **Agendavel** foi criada para garantir que todas as classes de serviço implementem o método **agendar(LocalDateTime dataHora)**. Isso assegura um padrão de comportamento para todas as operações de agendamento, mantendo o sistema coeso e organizado.

- **Interface Gráfica (JOptionPane):**
  Utilizamos **JOptionPane** para entrada e exibição de dados, o que proporciona uma interface mais intuitiva e de fácil uso para o usuário.

- **Tratamento de Erros (try-catch):**
  A implementação do sistema conta com blocos **try-catch** para capturar e tratar exceções de entrada inválida ou problemas no processamento, garantindo que o programa não seja interrompido abruptamente. Nesse projeto utilizamos para manter o controle de entrada de datas e horas.