dev Senior Python

RETO 2

Gestión de Veterinaria usando Programación Orientada a Objetos (Terminal)

Reto 2 - Gestión de Veterinaria usando Programación Orientada a Objetos (Terminal)

Contexto del Reto

La veterinaria

"Huella Feliz" es un pequeño negocio familiar que ha estado funcionando en la comunidad durante varios años. Debido al crecimiento en el número de clientes y al aumento de la demanda de servicios, los propietarios han decidido modernizar la gestión de la veterinaria con un sistema informático que les permita registrar a sus clientes y sus mascotas, programar citas, y llevar un control del historial de servicios prestados. Actualmente, esta información se gestiona de manera manual, lo que ha generado problemas como:

- Pérdida de información sobre clientes y sus mascotas.
- Dificultad para organizar y recordarcitas.
- Falta de un historial claro de los servicios prestados a cada mascota.

La Solicitud del Cliente

La dueña de "Huella Feliz", la señora Ana Torres, ha solicitado tu ayuda para desarrollar un sistema basado en consola que cubra las necesidades básicas de gestión de la veterinaria. Este sistema debe ser intuitivo, funcional y permitir a los empleados realizar todas las operaciones desde la terminal, ya que aún no están familiarizados con sistemas complejos.

Descripción del Reto

El sistema debe permitir gestionar las operaciones básicas de una veterinaria, tales como registrar clientes y mascotas, programar citas y gestionar un historial de servicios. Los estudiantes implementarán las funcionalidades en base a clases que representen los elementos del sistema.

Requerimientos Funcionales

1. GestióndeClientesyMascotas
    - Crear clientes con atributos como nombre, contacto y dirección.
    - Asociar una o más mascotas a cada cliente. Cada mascota tendrá atributos como nombre, especie, raza, edad y un historial de citas.

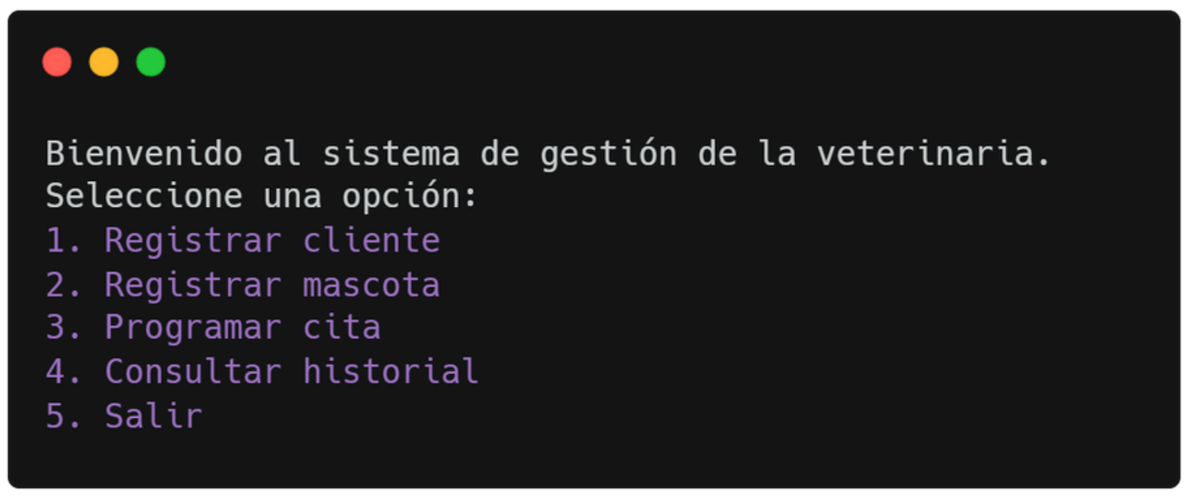
2. Gestión de Citas:
    - Registrar citas para una mascota con atributos como fecha, hora, servicio (consulta, vacunación, etc.) y veterinario asignado.
    - Actualizar o cancelar citas si es necesario.
3. Historial de Servicios:
    - Registrar servicios realizados a cada mascota.
    - Consultar el historial completo de servicios de una mascota.
4. Interacción Exclusivamente por Terminal:
    - El sistema debe ofrecer un menú interactivo que permita:
        - Registrar clientes y mascotas.
        - Programar y gestionar citas.
        - Consultar información de clientes, mascotas y servicios.
    - Toda la interacción debe realizarse mediante texto en la terminal.

Requerimientos Técnicos

1. Abstracción :
    - Diseñar clases que representen a los clientes, mascotas, servicios y citas.
2. Encapsulación:
    - Asegurar que los datos de las clases estén protegidos, proporcionando métodos para acceder y modificar los atributos.
3. Herencia :
    - Implementar herencia en las clases donde aplique (por ejemplo, una clase base para "Persona" de la que hereden "Cliente" y "Veterinario").
4. Polimorfismo:
    - Utilizar polimorfismo para que diferentes servicios puedan implementarse de forma específica pero con una interfaz común.

Interacción Terminal

- MenúPrincipal :
  - El programa debe iniciar con un menú en texto que permita al usuario seleccionar las acciones a realizar, como registrar clientes, programar citas o consultar el historial.
  - Ejemplo:



- Entrada y Validación:
  - Validar todas las entradas del usuario desde la terminal, asegurando que sean correctas (ejemplo: fechas en formato válido, nombres no vacíos, etc.).
  - Manejar errores con mensajes claros al usuario.

Rúbrica de Evaluación para el Reto

Criterio	Descripción	Puntos
Implementación Técnica (50 puntos)		
Simulaciónde datos y experimentación	El sistema permite registrar datos correctos y manejar la gestión de clientes, mascotas, citas y servicios correctamente.	15

Criterio	Descripción	Puntos
Análisis y procedimiento de datos	Permite al usuario realizar cálculos básicos o consultas detalladas en los historiales y citas de mascotas.	10
Validaciones y manejo de errores	Maneja adecuadamente entradas inválidas y asegura la estabilidad del programa.	10
Polimorfismo y herencia	Utiliza estos pilares de POO de manera correcta y funcional en el diseño del sistema.	15
Uso de Git y GitHub (20 puntos)		
Uso de ramas y trabajo colaborativo	El desarrollo del proyecto fue gestionado con ramas separadas y trabajo equitativo entre los miembros.	10
Commits regulares	commits son frecuentes y con mensajes descriptivos que reflejan los cambios realizados.	10
Presentación del Video (20 puntos)		
Explicación del proyecto y objetivos	Los estudiantes presentan adecuadamente el proyecto, explicando sus objetivos, funcionalidades y pilares aplicados.	5
Demostración de la funcionalidad	El video muestra una ejecución clara del programa, destacando las funcionalidades clave.	10
Claridad y equidad en la explicación	Ambos estudiantes participan en la presentación, alternando explicaciones con claridad y fluidez.	5
Documentacion 10 puntos		
Comentarios y claridad del código	El código está debidamente comentado y estructurado de manera comprensible.	5
README en GitHub	El repositorio contiene un README bien redactado con instrucciones para ejecutar el proyecto y una descripción general.	5

Puntuación Final :

- Excelente (90-100 puntos): Proyecto completamente funcional y bien presentado.
- Bueno (80-89 puntos): Proyecto funcional con detalles a mejorar en la implementación o presentación.
  Satisfactorio (70-79 puntos): Cumple la mayoría de los requisitos pero con deficiencias notables.
- Insuficiente (menos de 70 puntos): Presenta deficiencias significativas en funcionalidad o presentación.