


HISTORIA DE USUARIO

Nombre de la HU:		Uso de interfaces, depuración y manejo de excepciones
Objetivo de la HU		Como desarrollador del sistema de la clínica veterinaria, quiero aplicar interfaces, depuración y un manejo estructurado de errores en C#, para que el sistema sea más flexible, confiable y fácil de mantener.
TASK1	Comprender la diferencia entre clases abstractas e interfaces.	
	<ul style="list-style-type: none">• Revisar ejemplos prácticos de cuándo usar una clase abstracta y cuándo una interfaz.• Identificar en el sistema actual (pacientes, mascotas, servicios) casos donde una interfaz aporte mayor flexibilidad.• Documentar en comentarios las decisiones de diseño.	
TASK2	Implementar interfaces en escenarios reales de la clínica.	
	<ul style="list-style-type: none">• Crear una interfaz IRegistrable con el método Registrar().• Implementarla en Paciente y Mascota, asegurando que ambos puedan registrarse en el sistema de forma consistente.• Crear otra interfaz IAtendible con un método Atender(), e implementarla en clases de servicios veterinarios (ejemplo: ConsultaGeneral, Vacunacion).	
TASK3	Usar múltiples interfaces para lograr mayor flexibilidad.	
	<ul style="list-style-type: none">• Definir una interfaz INotificable con el método EnviarNotificacion().• Implementarla en Paciente para simular un recordatorio de cita.• Demostrar cómo una clase puede implementar varias interfaces al mismo tiempo (ejemplo: un Paciente puede ser IRegistrable y INotificable).	
T	Aplicar técnicas de depuración en el entorno de desarrollo.	

	 <ul style="list-style-type: none"> Colocar <i>breakpoints</i> estratégicos en el código para analizar la ejecución paso a paso. Inspeccionar valores de variables en tiempo de ejecución. Identificar un error forzado (ejemplo: división entre cero) y depurarlo con el IDE.
TASK 5	Aplicar un manejo estructurado de excepciones.
	<ul style="list-style-type: none"> Utilizar bloques try-catch-finally para manejar errores comunes en entradas de usuario. Crear excepciones personalizadas, por ejemplo MascotaNoEncontradaException, para manejar casos específicos. Implementar buenas prácticas como no silenciar excepciones y mostrar mensajes claros al usuario.
TASK 6	Agregar un sistema de registro de errores (logging básico).
	<ul style="list-style-type: none"> Guardar mensajes de error en un archivo de texto o mostrarlos en consola con formato adecuado. Documentar cómo este registro ayudaría en un entorno real para dar soporte técnico.
Criterios de aceptación. <ul style="list-style-type: none"> El sistema incluye interfaces implementadas correctamente en clases clave (Paciente, Mascota, servicios veterinarios). Se ha demostrado el uso de múltiples interfaces en una misma clase. Se han utilizado <i>breakpoints</i> y técnicas de depuración para analizar el flujo del programa. Los errores son manejados mediante bloques try-catch-finally. Existen excepciones personalizadas que representan casos específicos del dominio de la clínica. El sistema registra errores de manera básica, permitiendo su seguimiento. El código está organizado, con comentarios que explican las decisiones de diseño y buenas prácticas aplicadas. 	
History points: 20 puntos	
Cierre de actividad:	



Al finalizar esta cuarta semana, serás capaz de implementar interfaces para lograr una arquitectura más flexible, aplicar técnicas de depuración para encontrar errores y utilizar un manejo estructurado de excepciones que brinde estabilidad al sistema. Habrás creado excepciones personalizadas, practicado con *breakpoints* y agregado un registro de errores, lo que te permitirá escribir aplicaciones más confiables, profesionales y fáciles de mantener.