PROYECTO:
SISTEMA DE GESTIÓN DE CONTACTOS

INTRODUCCIÓN

El proyecto Sistema de Gestión de Contactos consiste en el desarrollo de una aplicación de consola en Python para la gestión de contactos personales. El sistema permite agregar, editar, eliminar, buscar y visualizar contactos, incorporando validaciones estrictas para asegurar la calidad de los datos ingresados.

El proyecto fue desarrollado en Python utilizando Programación Orientada a Objetos, estructuras de datos y realizando pruebas unitarias para las principales funcionalidades, de manera de garantizar el correcto funcionamiento.

Este proyecto se enmarca dentro de las actividades contenidas en el curso Fundamentos de Ciencia de Datos.

ESTRUCTURA DEL PROYECTO

- **proyecto_contactos/contacto.py:** Define la clase Contacto con validaciones para nombre, teléfono y email.
- **proyecto_contactos/gestorcontactos.py:** Define la clase GestorContactos para manejar la lista de contactos y operaciones CRUD.
- **proyecto_contactos/main.py:** Punto de entrada que ejecuta el menú principal. Contiene la interfaz de usuario basada en consola con menú interactivo.
- **proyecto_contactos/test_sistema_contactos.py:** Archivo de pruebas unitarias que verifica la funcionalidad y validaciones del sistema.

ASPECTOS TÉCNICOS

Validaciones Implementadas:

- **Nombre:** no puede estar vacío.
- **Teléfono:** Debe contener sólo dígitos y tener al menos 7 caracteres.
- **Email:** debe contener @ y un dominio válido, verificado mediante expresiones regulares.

Arquitectura

- **Modular:** Separación clara entre módulo, lógica de gestión y presentación.
- **Escalable:** Fácil de extender con persistencia o interfaz gráfica.
- **Robusta:** Manejo de errores mediante excepciones y mensajes con íconos para mejorar la experiencia del usuario.

PRUEBAS UNITARIAS

Cobertura de Pruebas

- Creación de contactos válidos e inválidos.
- Validación de setters.
- Operaciones CRUD en el gestor.
- Búsqueda por nombre y teléfono.
- Formato de salida con __str__.

Resultados

- Total de pruebas ejecutadas: 25
- Pruebas exitosas: 25
- Pruebas fallidas: 0

Las pruebas confirman que el sistema cumple con los requisitos funcionales y de validación.

DESAFÍOS ENFRENTADOS

- Evitar que se ingresen contactos con información incorrecta (validación datos de entrada).
- Resolver cómo capturar errores en la interfaz de usuario sin interrumpir el flujo del programa ni confundir al usuario.
- Separar responsabilidades sin complicar la integración (diseño modular).
- Asegurar que todas las funcionalidades (validaciones, CRUD, búsquedas, formato de salida) funcionen correctamente.

SOLUCIONES IMPLEMENTADAS

- Se añadieron validaciones en la clase Contacto usando expresiones regulares para emails, reglas de longitud para teléfonos, además de rechazar nombres vacíos.
- Uso de try/except para capturar errores de validación.
- Incorporación de íconos en los mensajes para mejorar la comunicación.
- Separación de lógica en módulos para facilitar pruebas y mantenimiento.
- Implementación de Pruebas unitarias exhaustivas para asegurar la calidad.

LOGROS OBTENIDOS

- Sistema funcional y estable.
- Validaciones completas en la clase Contacto.
- Interfaz amigable con mensajes claros.
- Pruebas unitarias con cobertura total.
- Documentación técnica y guía de ejecución.

CONCLUSIONES FINALES

El presente proyecto permitió aplicar los conocimientos en Python y Programación Orientada a Objetos para el desarrollo de una aplicación que cumple con las funciones básicas de gestión de contactos, asegurando la validez de los datos mediante reglas estrictas de validación, lo que garantiza que la información almacenada sea útil y confiable.

El diseño modular permitió contar con código ordenado, fácil de mantener y escalable. Por su parte, la incorporación de pruebas unitarias detalladas asegura que cada componente funcione correctamente y que los errores se detecten antes de llegar al usuario final.

En síntesis, este desarrollo no sólo resolvió el problema planteado sino que también dejó aprendizajes en términos de la aplicación de principios sólidos de ingeniería de software para obtener un sistema robusto, claro y confiable.