Evaluación del módulo

Proyecto: Gestor Inteligente de Clientes (GIC)

Situación inicial 📍

Unidad solicitante: Empresa SolutionTech (Startup de tecnología)

La empresa SolutionTech ha detectado la necesidad de mejorar su sistema de gestión de clientes mediante una solución escalable e integrable con otros servicios. Actualmente, los datos de los clientes se administran manualmente, lo que ha derivado en ineficiencias, duplicación de datos y problemas de seguridad.

La propuesta consiste en desarrollar una **plataforma integral de gestión de clientes** en **Python**, basada en **POO** y capaz de manejar distintos tipos de clientes, validaciones avanzadas, persistencia de datos y una interfaz de usuario básica.

Nuestro objetivo 📋

Desarrollar una plataforma en **Python** basada en **POO**, que permita **gestionarclientes, realizar validaciones** , asegurando una estructura modular, escalable y segura.

Objetivos específicos:

- Diseñar un **modelo UML** detallado del sistema.
- Implementar un sistema de gestión de clientes con **herencia, polimorfismo y encapsulación**.
- Incorporar validaciones avanzadas y **manejo de errores estructurado**.
- Implementar **persistencia** con **archivos JSON**.
- Uso de Try-Except
- Uso de Librerías
- Implementar **pruebas unitarias** para validar la funcionalidad.

Requerimientos

Funcionalidades esenciales:

- **Gestión de clientes:** Creación, edición y eliminación de clientes.
- **Diferenciación de tipos de clientes: ClienteRegular, ClientePremium, ClienteCorporativo.**
- **Manejo avanzado de atributos:** Validaciones de email.
- **Persistencia de datos:** Uso de **JSON**.
- **Registro de actividad:** Guardar logs de operaciones realizadas en el sistema.

Especificaciones técnicas:

- Implementación en **Python 3**.
- Aplicación de **clases, herencia y polimorfismo**.
- Validaciones avanzadas y **gestión de errores robusta**.

¿Qué vamos a validar? 🔍

- Estructura del código y principios de POO.
- Calidad del modelo UML y su correspondencia con la implementación.
- Uso adecuado de excepciones y validaciones.
- Persistencia de datos funcional y segura.

Entregables ✅

1. Paradigma de Orientación a Objetos

- Documento explicativo sobre **POO y su importancia en sistemas escalables**.
- Implementación de la **clase Cliente** con atributos básicos.
- Ejemplo práctico de **instanciación y uso de métodos**.

2. Programación Orientada a Objetos en Python

- Desarrollo de **clases y métodos** aplicando encapsulación.
- Implementación de validaciones en atributos.
- Creación de **métodos especiales (`__str__`, `__eq__`)**.

3. Diagramas de Clase

- Creación del **modelo UML** detallado con relaciones entre clases.
- Representación de **composición, herencia y agregación**.
- Uso de herramientas UML para documentación.

4. Herencia y Polimorfismo

- Implementación de subclases con **diferentes tipos de clientes**.
- Aplicación de **métodos sobrescritos y polimórficos**.
- Uso del **método `super()`** para reutilización de código.

5. Manejo de Errores y Excepciones

- Implementación de **excepciones personalizadas y logs de errores**.
- Validación de datos ingresados en la interfaz.
- Manejo de errores en la conexión a bases de datos.

6. Persistencia de Datos e Integración con APIs

- Implementación de **JSON** para almacenamiento.
- Implementación de pruebas unitarias.