Caso de Estudio: Sistema de Gestión de Tareas

Personales

Se necesita desarrollar un sistema simple para que un usuario pueda gestionar sus tareas diarias. El sistema debe permitir crear tareas, marcarlas como completadas y ver el estado general de sus pendientes.

Instrucciones:

* Lea cuidadosamente cada sección y pregunta.
* Responda de manera clara y concisa.
* Complete el código incompleto y corrija los errores. ● Justifique brevemente sus decisiones.

Requerimientos:

* Tareas: Cada tarea tiene una descripción, una fecha\_limite y un estado (pendiente/completada).
* Usuario: Un usuario puede tener múltiples tareas.

Funcionalidades:

* Crear nuevas tareas.
* Marcar tareas como completadas.
* Listar todas las tareas.
* Listar tareas pendientes.

# Parte 1: Diseño y Análisis de Clases

Se ha iniciado el diseño de la clase Tarea, pero está incompleta y puede contener errores.

class Tarea: def \_\_init\_\_(self, descripcion, fecha\_limite):

self.descripcion = descripcion self.fecha\_limite = fecha\_limite self.estado = "pendiente" self.\_\_id\_tarea = 1 def marcar\_completada(self): self.estado = "completada" print(f"Tarea '{self.descripcion}' marcada como completada.") def esta\_pendiente(self):

return self.estado == "pendiente" def \_\_str\_\_(self):

return f"Tarea: {self.descripcion} (Fecha Límite: {self.fecha\_limite}, Estado: {self.estado})"

## 1.1. Análisis y Corrección de la Clase Tarea

Preguntas y Tareas:

1. El atributo \_\_id\_tarea está inicializado con un valor fijo. ¿Qué problema fundamental presenta esto en un sistema donde se crearán múltiples tareas?

Proponga una mejora e implementa la misma en el código.

GENERA UNA FORMA UNICA DE IDENTIFICACION

1. En la clase Tarea, el atributo estado es público. ¿Qué problema podría presentar? Propone una mejora e implementa la misma en el código fuente.
2. CUALQUIER QUE TENGA EL CODIGO PUEDEN INGRESAR

## 1.2. Implementación de la Clase GestorTareas

Cree una nueva clase llamada GestorTareas que se encargue de manejar una colección de objetos Tarea.

1. Implemente el constructor para inicializar una lista vacía donde se almacenarán las tareas.
2. Implemente el método agregar\_tarea(self, tarea) que reciba un objeto Tarea y lo añada a la lista interna del gestor.
3. Implemente el método listar\_todas\_tareas(self) que imprima la descripción y el estado de todas las tareas.

**Nota. No utilizar conexión a base de datos. Resolver con estructuras de datos.**

## Parte 2: Persistencia de Datos

### 2.1. Diseño de la Base de Datos

1. Realice el modelo relacional para el sistema de gestión de tareas.
2. Escriba las sentencias sql para crear las tablas definidas previamente.
3. Escriba una sentencia sql para listar las tareas pendientes de un usuario en particular.