UNIVERSIDAD NACIONAL DE COLOMBIA

Universidad Nacional de Colombia - Sede Bogotá
Facultad de Ingeniería
Departamento de Sistemas e Industrial

Enunciado del proyecto experimental

Se les entrega un proyecto base desarrollado en Python que implementa un gestor sencillo de tareas. El proyecto incluye:

- La clase **Tarea**, que representa una tarea con los atributos:
  - **id:** identificador único (entero)
  - **nombre:** nombre de la tarea (string)
  - **estado:** indica si la tarea está pendiente o completada (string)

- La clase **GestorTareas**, que administra un conjunto de tareas y permite:
  - Agregar tareas
  - Marcar tareas como completadas.
  - Listar todas las tareas.

NOTA: En el código YA existen pruebas unitarias básicas incluidas (por ejemplo, prueba de agregar tarea y prueba de marcar como completada).

El código inicial es funcional y ha pasado las pruebas incluidas. El proyecto se entrega como un repositorio listo para clonar y empezar a trabajar.

RETO:

A partir de la hora de inicio tienes un tiempo máximo para el desarrollo de **80 minutos (1 hora y 20 minutos**)

Debes analizar el código base y extender su funcionalidad cumpliendo con los siguientes requisitos:

1. **Atributos nuevos**

Debes modificar la clase **Tarea** con los siguientes atributos nuevos:

- **prioridad:** nivel de prioridad de una tarea, Los valores permitidos son “Alta”, “Media”, “Baja”, “No aplica”, por defecto “Media” (string)
- **Fecha_limite**: fecha límite para realizar una tarea (datetime.date)
  El valor de fecha_limite se calcula automáticamente al crear la tarea según la prioridad, y no debe modificarse manualmente, siendo:
  a. High: 2 días
  b. Media: 5 días
  c. Baja: 10 días
  d. No aplica: None

Si no se especifica el valor de los nuevos atributos se debe utilizar los de defecto.

2. **Funcionalidades**

Debes implementar los siguientes dos métodos en la clase **GestorTareas**:

```
def listar_por_prioridad(self) -> List[Tarea]:
    """
    Devuelve una lista de tareas ordenadas por prioridad:
    Alta primero, luego Media, luego Baja.
    Mantiene el orden de alfabetico ascendente (a, b, c…)  dentro de cada grupo.
    """

def listar_vencidas(self, hoy: date) -> List[Tarea]:
    """
    Devuelve una lista de tareas cuya fecha límite es anterior al día indicado por defecto "hoy"
    """
```

3. **Requisitos**

- Todo el código debe ser legible, modular y seguir buenas prácticas de Python.
- Las pruebas unitarias deben cubrir las nuevas funcionalidades.
- **Las pruebas originales deben seguir pasando sin modificaciones.**
- Debes usar `unittest` y dejar comentarios que indiquen la prueba realizada.

Entrega

Se debe subir a un drive proporcionado al iniciar el experimento por medio de correo electronico PARA SUBIRLO A LA CARPETA DRIVE se debe empaquetar el proyecto en un zip (windows, click derecho comprimir) **TENIENDO en cuenta de no incluir el venv** (carpeta de entornos), el zip no debe pesar más de un par de MB.

Se debe renombrar el zip antes de subirlo con su usuario institucional en mi caso sería:

oalvarezr.zip

Tu entrega debe incluir:

- Código fuente modificado.
- Archivo de pruebas.
- Archivo `README.md` modificado, con:
    - En este apartado indica si consideras que completaste todo el reto y si algo quedó pendiente (por ejemplo, un método, una prueba, algo que no alcanzaste a completar pero tenías planeado).
    - Breve explicación de tu solución (opcional, máximo 5 líneas).

⚠️ Consideraciones ⚠️

- El uso de inteligencia artificial generativa (por ejemplo, ChatGPT, GitHub Copilot) está **permitido o no permitido** según el grupo al que se te asigne. **Esto será indicado al inicio.** en ambos casos, se puede utilizar libremente la documentación oficial de python y sus librerías, así como foros conocidos como stackoverflow
- El objetivo principal es que el código funcione correctamente, tenga buena calidad y esté respaldado por pruebas.
- **El experimento es académico: tus respuestas no afectan tu calificación ni relación con la universidad.**