Informe de Implementación del Sistema de Gestión de Calificaciones

1. Descripción del Sistema

El sistema implementado es una **aplicación de escritorio desarrollada en Python utilizando la librería `tkinter`**, diseñada para gestionar calificaciones de alumnos. Permite a los usuarios cargar datos de alumnos y sus notas desde un archivo CSV o ingresarlos manualmente. Una vez cargados los datos, el sistema ofrece diversas funcionalidades para **generar reportes y ordenar la información**, facilitando el análisis y seguimiento del rendimiento académico.

La interfaz de usuario es interactiva y se compone de una ventana principal que muestra la grilla de datos de los alumnos, junto con una barra de menú que organiza las diferentes acciones disponibles:

- **Ingresar Calificación**: Permite cargar datos desde un archivo CSV o ingresar información de un nuevo alumno manualmente.
- **Elegir Reporte**: Ofrece opciones para generar distintos tipos de informes sobre los datos cargados.
- **Ordenar por**: Permite ordenar la grilla de datos según el nombre del alumno o su nota final.
- **Finalizar**: Para salir de la aplicación.

2. Estructura de Datos Utilizada

El sistema utiliza principalmente una **lista global llamada `current_grid_data`** para almacenar todos los registros de alumnos. Cada registro de alumno es representado como una **tupla de 6 elementos (strings)**, siguiendo el formato:

```
("Alumno", "Materia", "Nota 1", "Nota 2", "Nota 3", "Nota Final")
```

Por ejemplo, un registro podría ser: `("Juan Pérez", "Matemáticas", "7", "8", "6", "7.0")`.

Ventajas de usar tuplas en este contexto:

- **Inmutabilidad**: Una vez que se crea un registro de alumno (tupla), sus valores no pueden ser modificados. Esto es adecuado para datos que se cargan y luego se consultan o reportan, asegurando que los datos originales no sean alterados accidentalmente.
- **Eficiencia**: Las tuplas son generalmente más eficientes en términos de memoria y rendimiento que las listas para colecciones de elementos fijos.

La elección de una lista de tuplas permite:

- Almacenar múltiples registros de alumnos de manera organizada.

- Acceder a los datos por índice para realizar operaciones de ordenamiento y filtrado.
- Facilitar la visualización de los datos en una grilla tabular.

3. Resumen de Cada Función

A continuación, se detalla la funcionalidad de cada función implementada en el programa:

Funciones de Reporte:

- **`mostrar_reporte(title, contenido)`**:
  - **Descripción**: Crea y muestra una nueva ventana (`tk.Toplevel`) que contiene el texto de un reporte generado.
  - **Uso**: Es una función auxiliar utilizada por todas las demás funciones de reporte para presentar sus resultados al usuario de manera legible.
- **`listado_alumnos()`**:
  - **Descripción**: Genera un reporte que muestra el listado completo de todos los alumnos, junto con sus materias, notas individuales y nota final.
  - **Lógica**: Itera sobre `current_grid_data` y formatea cada fila para incluirla en el reporte.
- **`promedio_por_materia()`**:
  - **Descripción**: Calcula y muestra el promedio de las notas finales para cada materia cargada en el sistema.
  - **Lógica**: Utiliza un diccionario para agrupar las notas finales por materia y luego calcula el promedio para cada una.
- **`alumnos_mayor()`**:
  - **Descripción**: Permite al usuario ingresar un valor numérico y luego genera un reporte de los alumnos cuya nota final es superior a ese valor.
  - **Lógica**: Abre un diálogo (`tk.Toplevel`) para solicitar el valor, valida la entrada y filtra `current_grid_data` para encontrar y mostrar los alumnos que cumplen con la condición.
- **`alumnos_menor()`**:
  - **Descripción**: Genera un reporte de los alumnos que tienen al menos una nota (Nota 1, Nota 2 o Nota 3) menor a 4.
  - **Lógica**: Itera sobre los datos y verifica las notas individuales de cada alumno.
- **`apro_desaprobado()`**:
  - **Descripción**: Calcula y muestra la cantidad de alumnos aprobados (nota final >= 6) y desaprobados (nota final < 6) por cada materia.
  - **Lógica**: Agrupa los alumnos por materia y cuenta aprobados y desaprobados basándose en la nota final.

Funciones de Gestión de Grilla y Datos:

- **`encab_grid(frame)`**:
  - **Descripción**: Crea las etiquetas de encabezado para la grilla de visualización de datos en la interfaz.

- **Uso**: Es llamada al inicio para configurar la grilla y cada vez que se recarga la grilla.

- **`load_data_into_grid(frame, data)`**:
  - **Descripción**: Carga y muestra los datos proporcionados en la grilla (`tk.Frame`) de la interfaz de usuario. Limpia cualquier contenido previo antes de cargar los nuevos datos.
  - **Lógica**: Itera sobre la lista de datos y crea `tk.Label` para cada celda, organizándolos en la grilla.
- **`cargarDatosArchivo()`**:
  - **Descripción**: Abre un cuadro de diálogo para que el usuario seleccione un archivo CSV, lee su contenido y carga los datos de los alumnos en `current_grid_data` y en la grilla.
  - **Lógica**: Utiliza `filedialog` para la selección, lee el archivo línea por línea, divide cada línea por comas y agrega las tuplas resultantes a `current_grid_data`. Incluye manejo de errores para archivos y formato.
- **`abrir_dialogo_ingreso_manual()`**:
  - **Descripción**: Abre una nueva ventana de diálogo que permite al usuario ingresar manualmente los datos de un nuevo alumno (nombre, materia, Nota 1, Nota 2, Nota 3).
  - **Lógica**: Crea campos de entrada (`tk.Entry`), valida que los datos sean correctos (ej. notas numéricas y en rango), calcula la nota final y añade el nuevo registro a `current_grid_data`, actualizando la grilla.

Funciones de Ordenamiento:

- **`insertion_sort(arr, key_index, reverse=False, is_numeric=False)`**:
  - **Descripción**: Implementa el algoritmo de ordenamiento por inserción. Permite ordenar una lista de tuplas según un índice específico (columna), en orden ascendente o descendente, y con la opción de tratar los valores como numéricos para una comparación adecuada.
  - **Algoritmo**: Funciona comparando un elemento con los elementos ya ordenados a su izquierda y insertándolo en la posición correcta. Es eficiente para conjuntos de datos pequeños o casi ordenados.
- **`ordenarPorNombre()`**:
  - **Descripción**: Ordena los datos de los alumnos en `current_grid_data` alfabéticamente por el nombre del alumno (columna 0) utilizando `insertion_sort`.
  - **Lógica**: Llama a `insertion_sort` con `key_index=0`, `reverse=False` y `is_numeric=False`.
- **`ordenarPorNota()`**:
  - **Descripción**: Ordena los datos de los alumnos en `current_grid_data` por la Nota Final (columna 5) de mayor a menor, utilizando `insertion_sort`.
  - **Lógica**: Llama a `insertion_sort` con `key_index=5`, `reverse=True` y `is_numeric=True`.

---

4. Organización del Trabajo Grupal

- **Definición de Requisitos y Diseño (Todos)** Nos reunimos siempre en forma virtual para analizar tareas, dividir consignas y responsabilidades de cada integrante del equipo.

- **Nicolás Celentano**:
    - Creación de la interfaz de usuario, ordenamiento y la carga manual y de archivos
    - Creación de la ventana principal y la barra de menú con `tkinter`.
    - Implementación del `tk.Frame` para la grilla y las funciones `encab_grid` y load_data_into_grid para la visualización de datos.

- **Bautista Rossi**:
    - Pruebas, corrección de los errores que aparecían enumerar y comentarios.
    - Informes y reportes.
    - Mejoras en la interfaz de usuario