

Universidad Rafael Landívar  
Facultad de Ingeniería  
Licenciatura en Ingeniería en Informática y Sistemas

## **Documentación - Gestor de Actividades Universitarias**

Curso: Programación Avanzada  
Sección: 03

Integrantes:

Mario Rolando Matheu Duque – 1567324

María Belén Mendoza García - 1517225

Guatemala, agosto 2025

## ÍNDICE

INTRODUCCIÓN: .....	3
OBJETIVOS:.....	3
Objetivo General.....	3
Objetivos Específicos.....	3
ANÁLISIS Y DISEÑO:.....	4
DESARROLLO DEL PROYECTO: .....	4
MANUAL DE USUARIO: .....	5
Ingreso al sistema.....	5
Al ejecutar el programa, se mostrará un menú con las siguientes opciones: .....	5
Recomendaciones de uso .....	11
TABLA DE RESPONSABILIDADES Y EVIDENCIAS: .....	11
PROBLEMAS ENCONTRADOS: .....	13
DECISIONES DE DISEÑO:.....	13
ANEXOS: .....	15
Pseudocódigo del proyecto: .....	15
Diagrama UML del sistema:.....	19

## **INTRODUCCIÓN:**

El presente documento posee distintos recursos gráficos que buscan ayudar a entender mejor el proyecto. Incluye el pseudocódigo del sistema, imágenes de la interfaz y del proceso de realización del proyecto y ejemplos de cómo usarlo, mostrando las funciones principales. Su objetivo es ofrecer apoyo visual y técnico para comprender el proyecto “Plataforma de Gestión y Evaluación de Cursos Online”.

El proyecto se basó en lo siguiente:

Un sistema diseñado para gestionar cursos online, usuarios (estudiantes e instructores) y evaluaciones.

Las principales funciones del programa son:

- Crear y administrar cursos (nombre, código, instructor).
- Registrar usuarios (estudiantes e instructores).
- Inscribir estudiantes en cursos.
- Crear evaluaciones (exámenes, tareas) para cada curso.
- Registrar calificaciones por estudiante y evaluación.
- Consultar cursos, estudiantes inscritos, evaluaciones y calificaciones.
- Generar reportes simples (ej: estudiantes con promedio bajo).

## **OBJETIVOS:**

### **Objetivo General**

Desarrollar un sistema que permita gestionar cursos en línea, usuarios y evaluaciones de manera sencilla y organizada, aplicando los principios de la programación orientada a objetos en Python.

### **Objetivos Específicos**

- Crear y administrar cursos con nombre, código e instructor.
- Registrar usuarios (estudiantes e instructores) con sus datos principales.
- Inscribir estudiantes en cursos y mantener listas actualizadas.
- Diseñar evaluaciones y registrar calificaciones.
- Consultar información sobre cursos, estudiantes inscritos y reportes de notas.
- Aplicar herencia, polimorfismo y manejo de errores para asegurar robustez en el sistema.
- Utilizar Git y GitHub para un trabajo en equipo ordenado y con control de versiones.

## ANÁLISIS Y DISEÑO:

Para el diseño del sistema se definieron las siguientes clases principales:

- Persona: Clase base que contiene atributos generales como ID y nombre.
- Estudiante (hereda de Persona): Permite inscribir cursos y almacenar calificaciones.
- Profesor (hereda de Persona): Puede crear cursos y evaluaciones.
- Curso: Incluye nombre, código, profesor, estudiantes inscritos y evaluaciones.
- Evaluación: Representa tareas o exámenes, con fecha, tipo, nota máxima y calificaciones.
- Sistema de Gestión de Cursos: Clase central que coordina estudiantes, profesores, cursos y evaluaciones.

Se elaboró un diagrama UML para representar las relaciones y herencias entre estas clases. Además, se escribió pseudocódigo para funciones clave como inscribir estudiantes, crear cursos, registrar evaluaciones y calificaciones.

## DESARROLLO DEL PROYECTO:

El desarrollo se realizó en Python utilizando programación orientada a objetos. Se aplicaron conceptos de encapsulamiento, herencia y polimorfismo, junto con estructuras de datos como listas y diccionarios.

El trabajo en equipo se organizó mediante un repositorio en GitHub, donde cada integrante aportó código, documentación y pruebas. Se utilizaron issues con etiquetas, ramas y commits individuales para mantener un control claro del avance.

**Repositorio en GitHub:** <https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online.git>



## MANUAL DE USUARIO:

El sistema de gestión de cursos funciona a través de un menú interactivo en consola. A continuación, se describe el uso de cada opción:

### Ingreso al sistema

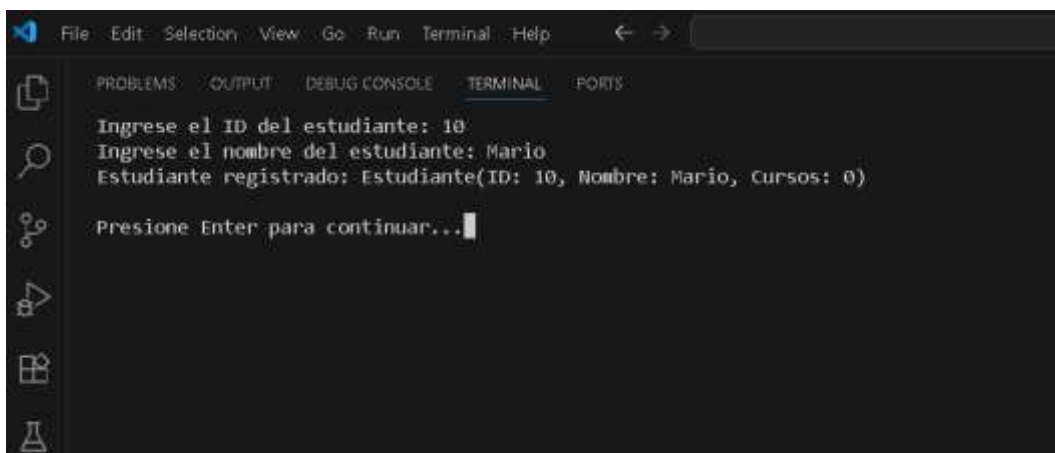
Al ejecutar el programa, se mostrará un menú con las siguientes opciones:

```
=====
Sistema de Gestión de Cursos
=====
1. Registrar Estudiante
2. Registrar Profesor
3. Crear Curso
4. Inscribir Estudiante en Curso
5. Crear Evaluación
6. Registrar Calificación
7. Obtener Promedio Estudiante
8. Listar Estudiantes con Promedio Bajo
9. Listar Estudiantes con Promedio Alto
10. Listar Cursos
11. Listar Estudiantes
12. Listar Evaluaciones de un Curso
13. Listar Profesores
14. Listar Estudiantes de un Curso
0. Salir

Seleccione una opción: █
```

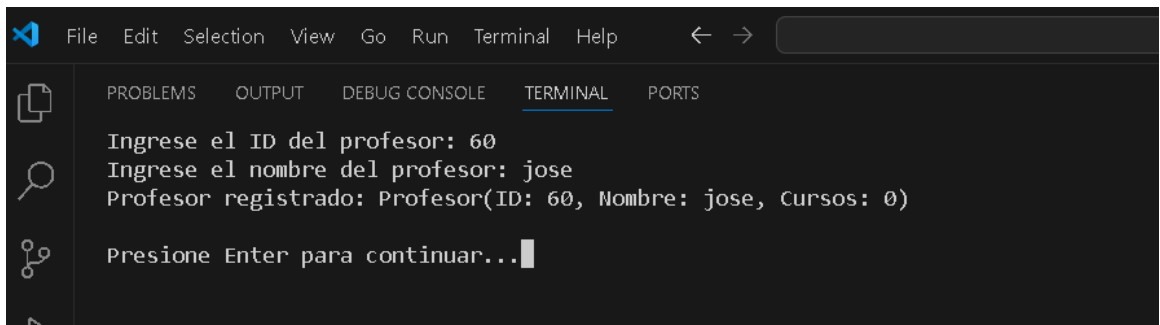
#### 1. Registrar estudiante

- Permite ingresar un nuevo estudiante proporcionando su ID y nombre.
- El sistema validará que no exista otro estudiante con el mismo ID.



## 2. Registrar profesor

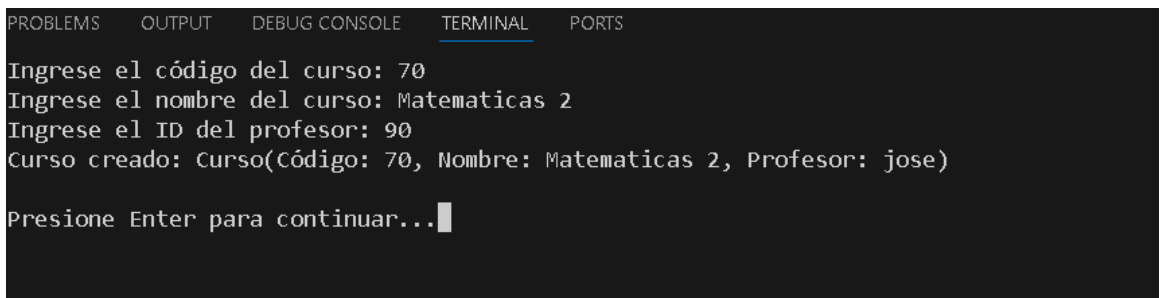
- Permite registrar un profesor ingresando ID y nombre.
- Se valida que no se repita el ID.



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Ingrese el ID del profesor: 60
Ingrese el nombre del profesor: jose
Profesor registrado: Profesor(ID: 60, Nombre: jose, Cursos: 0)
Presione Enter para continuar...
```

## 3. Crear curso

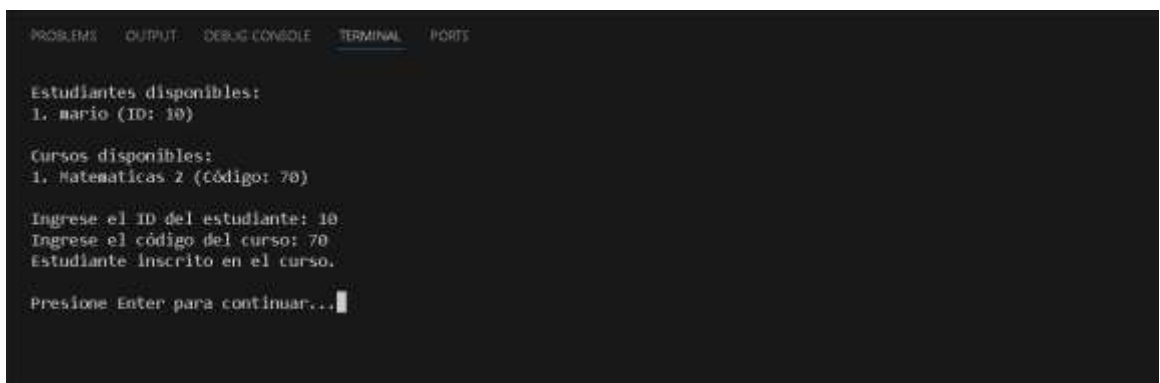
- Se debe ingresar el código y nombre del curso.
- Se asigna un profesor previamente registrado.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Ingrese el código del curso: 70
Ingrese el nombre del curso: Matematicas 2
Ingrese el ID del profesor: 90
Curso creado: Curso(Código: 70, Nombre: Matematicas 2, Profesor: jose)
Presione Enter para continuar...
```

## 4. Inscribir estudiante en curso

- Muestra la lista de estudiantes y cursos disponibles.
- Permite seleccionar un estudiante y asignarlo a un curso.
- El sistema evita inscripciones duplicadas.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Estudiantes disponibles:
1. mario (ID: 10)

Cursos disponibles:
1. Matematicas 2 (código: 70)

Ingrese el ID del estudiante: 10
Ingrese el código del curso: 70
Estudiante inscrito en el curso.

Presione Enter para continuar...
```

## 5. Crear evaluación

- Permite que un profesor cree exámenes o tareas.
- Se ingresan datos como nombre de la evaluación, tipo, fecha y puntaje máximo.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Cursos disponibles:
1. Matematicas 2 (Código: 70)

Ingrese el código del curso: 70
Ingrese el nombre de la evaluación: Ecuaciones diferenciales
Ingrese el tipo de evaluación (examen/tarea): tarea
Ingrese la fecha límite (YYYY-MM-DD): 2025-04-03
Ingrese el puntaje máximo: 100
Ingrese el ID de la evaluación: 30
Evaluación creada: Evaluación(ID: 30, Nombre: Ecuaciones diferenciales, Tipo: tarea, Curso: 70)

Presione Enter para continuar...
```

## 6. Registrar calificación

- Se selecciona un curso, una evaluación y un estudiante inscrito.
- Se ingresa la calificación, validando que esté dentro del rango permitido.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Cursos disponibles:
1. Matematicas 2 (Código: 70)

Ingrese el código del curso: 70
1. Ecuaciones diferenciales (ID: 30)

Ingrese el ID de la evaluación: 30

Estudiantes inscritos en el curso:
1. Mario (ID: 10)

Ingrese el ID del estudiante: 10
Ingrese la calificación: 68
Calificación registrada.

Presione Enter para continuar...
```

### 7. Obtener promedio de estudiante

- Se ingresa el ID de un estudiante.
- Se puede calcular el promedio general o solo de un curso específico.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Estudiantes disponibles:
1. mario (ID: 10)

Ingrese el ID del estudiante: 10
Ingrese el código del curso (dejar vacío para promedio general):
Promedio del estudiante: 68.00

Presione Enter para continuar...
```

### 8. Listar estudiantes con promedio bajo

- Muestra estudiantes cuyo promedio esté por debajo de un límite definido (por defecto 60).

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Ingrese el código del curso (dejar vacío para todos los cursos):
Ingrese el límite de promedio (default 60): 60
No hay estudiantes con promedio bajo.

Presione Enter para continuar...
```

### 9. Listar estudiantes con promedio alto

- Muestra estudiantes con promedios superiores al límite (por defecto 90).

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Ingrese el código del curso (dejar vacío para todos los cursos):
Ingrese el límite de promedio (default 90): 55

Estudiantes con promedio alto:
  mario - Promedio: 68.00

Presione Enter para continuar...
```



#### 10. Listar cursos

- Presenta un listado de todos los cursos creados en el sistema.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Cursos registrados:
1. Matematicas 2 (Código: 70)

Presione Enter para continuar...█
```

#### 11. Listar estudiantes

- Presenta un listado de todos los estudiantes registrados.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Estudiantes registrados:
1. mario (ID: 10)

Presione Enter para continuar...█
```

#### 12. Listar evaluaciones de un curso

- Se selecciona un curso y el sistema muestra todas las evaluaciones creadas.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Cursos disponibles:
1. Matematicas 2 (Código: 70)

Ingresa el código del curso: 70

Evaluaciones del curso 70:
1. Ecuaciones diferenciales (ID: 30)

Presione Enter para continuar...█
```

### 13. Listar profesores

- Presenta todos los profesores registrados.

```
1  import os

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Profesores registrados:
1. jose (ID: 90)

Presione Enter para continuar...
```

### 14. Listar estudiantes de un curso

- Muestra los estudiantes inscritos en un curso específico.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Cursos disponibles:
1. Matematicas 2 (Código: 70)

Ingrese el código del curso: 70

Estudiantes inscritos en el curso 70:
1. mario (ID: 10)

Presione Enter para continuar...
```

### 15. Salir del sistema

- Finaliza la ejecución del programa.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Saliendo del sistema. ¡Hasta pronto!
Sistema de gestión de cursos ha sido eliminado
Estudiante mario (ID: 10) ha sido eliminado
Profesor jose (ID: 90) ha sido eliminado
Curso Matematicas 2 (Código: 70) ha sido eliminado
Evaluación Ecuaciones diferenciales (ID: 30) ha sido eliminada
PS C:\Users\mario>
```

### Recomendaciones de uso

- Verifique siempre que los IDs sean únicos para evitar conflictos.
- Registre primero profesores y estudiantes antes de crear cursos o evaluaciones.
- Guarde capturas de las salidas importantes como evidencia de pruebas.
- En caso de error de ingreso, el sistema muestra mensajes claros para guiar al usuario.

### TABLA DE RESPONSABILIDADES Y EVIDENCIAS:

Integrante	Carnet	Tarea Asignada	Evidencia en GitHub	Fecha de Entrega Parcial	Validación Docente
María Belén Mendoza Garcia	1517225	1. Configurar estructura de carpetas del proyecto	1. <a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/1">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/1</a>		
		3. Diseño de diagrama UML de clases #3	3. <a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/3">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/3</a>		
		4. Pseudocódigo de funciones principales #4	4. <a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/4">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/4</a>		
		5. Implementación de sistema de cursos #6	5. <a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/6">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/6</a>		
		6. Implementación de sistema de reportes #10	6. <a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/10">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/10</a>		
		7. Crear el menú interactivo #23	7. <a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/23">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/23</a>		

Integrante	Carnet	Tarea Asignada	Evidencia en GitHub	Fecha de Entrega Parcial	Validación Docente
Mario Rolando Matheu Duque	1567324	1.Pseudocódigo de funciones principales #4	<a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/4">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/4</a>		
		2.Implementación de sistema de usuarios #5	<a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/5">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/5</a>		
		3.Implementación de sistema de cursos #6	<a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/6">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/6</a>		
		4. Implementación de sistema de evaluaciones #7	<a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/7">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/7</a>		
		5.Implementación de sistema de calificaciones #9	<a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/9">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/9</a>		
		6.Implementación del Sistema Principal de Gestión #17	<a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/17">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/17</a>		
		7. Implementar args y kwargs #25	<a href="https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/25">https://github.com/mari-j1407b/Plataforma-de-Gestion-y-Evaluacion-de-Cursos-Online/issues/25</a>		

## PROBLEMAS ENCONTRADOS:

Durante el desarrollo se enfrentaron los siguientes retos:

- **Herencia y polimorfismo:** Al inicio costó definir qué atributos debían estar en la clase base y cuáles en las clases hijas. Se solucionó reorganizando el diseño UML para visualizar las clases de forma más clara antes de empezar a programar.
- **Errores de sintaxis:** Ocurrieron errores por puntos, comas o mayúsculas mal colocadas. Fueron corregidos mediante pruebas y revisiones al correr el código.
- **Inscripción duplicada:** Al inicio los estudiantes podían inscribirse más de una vez al mismo curso. Se resolvió con validaciones en el método `inscribir_curso`.
- **Calificaciones fuera de rango:** En algunos casos se podían registrar notas menores a 0 o mayores al máximo. Se agregó validación para corregirlo.
- **Trabajo en GitHub:** Hubo confusión con ramas y merges. Algunas veces ramas no mostraban el progreso realizado en la rama anterior, lo que impedía aprobar merges en GitHub de forma correcta. Se resolvió con investigación y asegurando que cada rama esté ligada a las ramas previas y no directamente al main, para poder visualizar correctamente el progreso del programa.

## DECISIONES DE DISEÑO:

- **¿Cómo se manejó la herencia y el polimorfismo en el diseño del programa? ¿Qué ventajas brindó?**

Se usó herencia en las clases estudiantes y profesor, que heredan de la clase base persona. Esto permite reutilizar atributos comunes como id y nombre, y solo añadir lo específico de cada uno (por ejemplo, cursos inscritos en el caso de estudiantes, y cursos impartidos en el caso de profesores).

El polimorfismo se aplica principalmente en los métodos `__str__`, ya que cada clase redefine este método para mostrar su propia información, aunque todos comparten la misma “firma” del método. Esto brinda flexibilidad y hace más fácil imprimir objetos de diferentes tipos sin preocuparnos por cómo mostrarán su información.

- **¿Qué estructuras de datos fueron más útiles y por qué?**

Listas y diccionarios, ya que las listas permitieron guardar colecciones de estudiantes, profesores, cursos y evaluaciones, y son útiles porque permiten recorrer y agregar elementos fácilmente; y los diccionarios se usaron en las calificaciones, porque permiten asociar rápidamente un ID de estudiante con sus notas, facilitando la búsqueda directa por clave.

- **¿Cómo organizaron su trabajo usando Git y GitHub?**

Se utilizó la creación de issues en GitHub para visualizar mejor todo lo que el programa requería, y a partir de ello se repartieron dichas issues según habilidades y preferencias de ambos miembros del equipo.

- **¿Qué mejoras se harían si hubiese más tiempo para realizar el proyecto?**

Estos parámetros se aplicaron en varios métodos para dar flexibilidad en la creación de objetos.

Podrían implementarse:

- Persistencia en archivos o base de datos para no perder los datos al cerrar el programa. (Por ejemplo, con un archivo .json)
- Una interfaz gráfica en lugar de solo consola.
- Reportes más completos, como históricos de calificaciones o ranking de estudiantes.

## ANEXOS:

### Pseudocódigo del proyecto:

A continuación, se muestra el pseudocódigo del programa, el cual refleja la estructura principal de su funcionamiento.

```
Clase Persona:                                # Clase base general para representar cualquier persona
    atributos: id, nombre                      # Cada persona tendrá un identificador único (id) y un nombre
    métodos:
        constructor(id, nombre)               # Crea una persona asignándole un id y un nombre
        mostrar como texto -> "Persona(ID, Nombre)" # Devuelve una cadena con los datos de la persona
        al eliminar -> mostrar mensaje        # Muestra un mensaje cuando el objeto persona es destruido

Clase Estudiante hereda Persona:              # Subclase que representa a los estudiantes
    atributos: cursos_inscritos (lista), calificaciones (diccionario)
                                                # Lista de cursos en los que está inscrito y sus notas en cada evaluación
    métodos:
        inscribir_curso(curso):                # Método para inscribir al estudiante en un curso específico
            si curso no está en cursos_inscritos: # Valida que no esté repetido
                agregar curso                    # Agrega el curso a la lista del estudiante
                agregar estudiante al curso      # Registra al estudiante dentro de la lista del curso
                retornar verdadero              # Confirma que la inscripción fue exitosa
            sino retornar falso                  # Si ya estaba inscrito, devuelve falso para evitar duplicidad
        mostrar como texto -> "Estudiante(ID, Nombre, CantidadCursos)"
                                                # Devuelve cadena con ID, nombre y número de cursos inscritos
        al eliminar -> mostrar mensaje          # Mensaje de confirmación cuando se elimina el estudiante

Clase Profesor hereda Persona:                # Subclase que representa a los profesores
    atributos: cursos_impardidos (lista) # Cursos que el profesor está enseñando actualmente
    métodos:
        crear_curso(sistema, nombre_curso, codigo): # Permite al profesor crear un curso nuevo
            si ya existe curso con ese código en sistema: # Verifica que no se repita el código
                mostrar error y retornar nulo # Error si ya existe curso con ese código
            crear nuevo curso                    # Crea el objeto curso
            registrar curso en sistema          # Añade el curso al sistema de gestión
            agregar curso a cursos_impardidos   # Asocia este curso con el profesor creador
            retornar curso                      # Devuelve el curso recién creado
        mostrar como texto -> "Profesor(ID, Nombre, CantidadCursos)"
                                                # Devuelve cadena con ID, nombre y cantidad de cursos impartidos
        al eliminar -> mostrar mensaje          # Mensaje al eliminar un profesor
```

```

Clase Curso:                                     # Clase que representa un curso académico
    atributos: nombre, código, profesor, estudiantes_inscritos, evaluaciones
                                                    # Contiene la info del curso, su profesor, alumnos y evaluaciones
    métodos:
        agregar_evaluacion(evaluación): # Agrega una evaluación (examen, tarea, etc.) al curso
            si ya existe evaluación con mismo nombre: # Valida que no esté repetida
                mostrar error y retornar falso # Error si la evaluación ya existe
            agregar evaluación                # Si no existe, se guarda en la lista
            retornar verdadero                # Retorna confirmación de éxito
        listar_evaluaciones:                  # Muestra todas las evaluaciones del curso
            si no hay evaluaciones -> mostrar mensaje # Si no tiene ninguna, muestra aviso
            sino listar todas                 # Si existen, se muestran todas en pantalla
            mostrar como texto -> "Curso(Código, Nombre, Profesor)"
                                                    # Devuelve un resumen con código, nombre y profesor
            al eliminar -> mostrar mensaje # Mensaje de confirmación cuando se elimina curso

Clase Evaluación:                               # Clase que representa una tarea, examen o actividad
evaluada
    atributos: id, nombre, fecha, nota_maxima, tipo, curso_codigo, calificaciones
                                                    # Tiene datos de la evaluación y las notas de los estudiantes
    métodos:
        registrar_calificacion(estudiante, calificacion):
                                                    # Registra la nota de un estudiante en la evaluación
            si calificación fuera de rango -> error # Valida que la nota esté entre 0 y la nota
máxima
            guardar calificación en evaluación y estudiante # Guarda la nota en ambas partes
            retornar verdadero                # Confirma que se registró la nota
            mostrar como texto -> "Evaluación(ID, Nombre, Tipo, Curso)"
                                                    # Devuelve info resumida de la evaluación
            al eliminar -> mostrar mensaje # Mensaje de confirmación al eliminar evaluación

```



```

Clase SistemaGestionCursos:           # Clase principal que administra todo el sistema académico
    atributos: estudiantes, profesores, cursos, evaluaciones
        # Contiene las listas globales de estudiantes, profesores, cursos y evaluaciones
    métodos:
        registrar_estudiante(id, nombre): # Registra un nuevo estudiante en el sistema
            si ya existe con ese id -> error # Verifica duplicado por ID
            crear nuevo estudiante y guardar # Si no existe, se agrega
        registrar_profesor(id, nombre): # Registra un nuevo profesor
            si ya existe con ese id -> error
            crear nuevo profesor y guardar
        registrar_curso(curso): # Registra un nuevo curso
            si ya existe con ese código -> error
            guardar curso
        crear_evaluacion(curso_id, nombre, tipo, fecha, puntaje, id_eval):
            # Crea evaluación asociada a un curso
            buscar curso # Localiza el curso
            si no existe -> error
            verificar que id_eval no exista # Evita repetir evaluaciones con mismo id
            crear evaluación # Si todo está bien, crea la evaluación
            si se agrega correctamente -> guardar en sistema
        inscribir_estudiante_en_curso(est_id, curso_id):
            # Inscribe un estudiante en un curso
            buscar estudiante y curso
            si alguno no existe -> error
            llamar a inscribir_curso # Usa el método del estudiante
        registrar_calificacion(eval_id, est_id, nota):
            # Registra la calificación de un estudiante
            buscar evaluación y estudiante
            si no existen -> error
            verificar que estudiante esté inscrito en curso # Validación
            registrar calificación # Si cumple, se asigna la nota
        obtener_promedio_estudiante(est_id, curso_id opcional):
            # Calcula promedio de un estudiante

```

```

    si se da curso -> promedio en ese curso
    sino -> promedio general
listar_cursos, listar_estudiantes, listar_profesores
    # Muestran listados de cursos, estudiantes y profesores
listar_evaluaciones_curso(curso_id)
    # Lista todas las evaluaciones de un curso
listar_estudiantes_curso(curso_id)
    # Lista todos los estudiantes de un curso
estudiantes_promedio_bajo(curso_id opcional, limite=60)
    # Devuelve estudiantes con promedio bajo (por debajo de 60
o límite definido)
estudiantes_promedio_alto(curso_id opcional, limite=90)
    # Devuelve estudiantes con promedio alto (90 o límite
definido)
al eliminar -> mostrar mensaje    # Mensaje al cerrar/eliminar sistema

Función MENU principal:            # Interfaz principal para interactuar con el sistema
crear objeto sistema                # Se crea la instancia del sistema

repetir hasta que usuario elija "0 - Salir":
    # Ciclo que mantiene el menú activo hasta que el usuario
salga
mostrar menú con opciones:        # Se imprimen las opciones disponibles
    1. Registrar estudiante
    2. Registrar profesor
    3. Crear curso
    4. Inscribir estudiante en curso
    5. Crear evaluación
    6. Registrar calificación
    7. Obtener promedio estudiante
    8. Listar estudiantes con promedio bajo
    9. Listar estudiantes con promedio alto
    10. Listar cursos
    11. Listar estudiantes
    12. Listar evaluaciones de un curso
    13. Listar profesores
    14. Listar estudiantes de un curso
    0. Salir

```

según opción ingresada:

llamar al método correspondiente del sistema # Se ejecuta la función correspondiente

mostrar resultados o mensajes de error # Se imprime el resultado al usuario

### Diagrama UML del sistema:

A continuación, se muestra el diagrama UML del sistema, que permite visualizar de forma más clara la distribución de las clases del programa.

