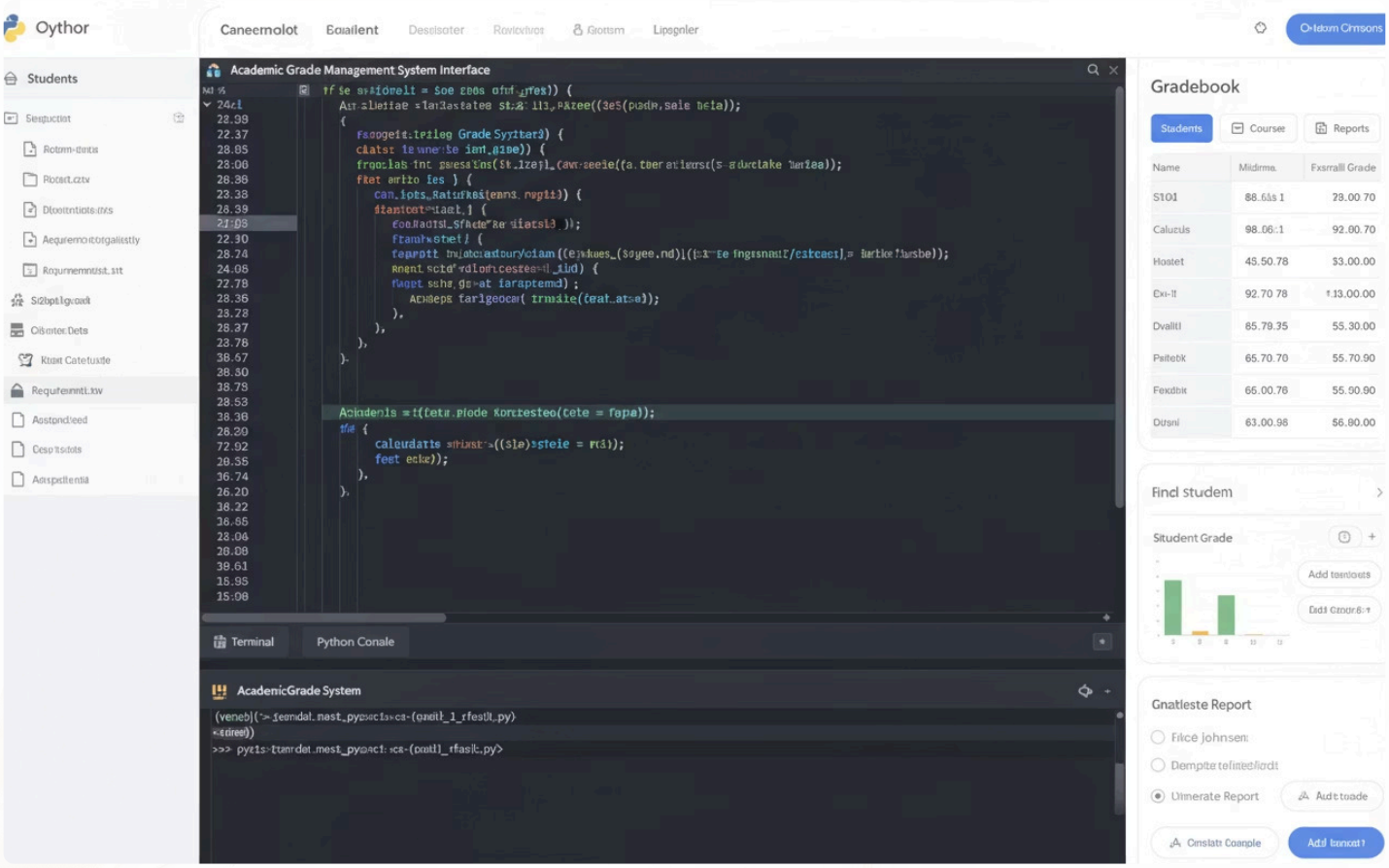
Manual Técnico – Gestor de Notas Académicas



Nombre del proyecto: Gestor de Notas Académicas

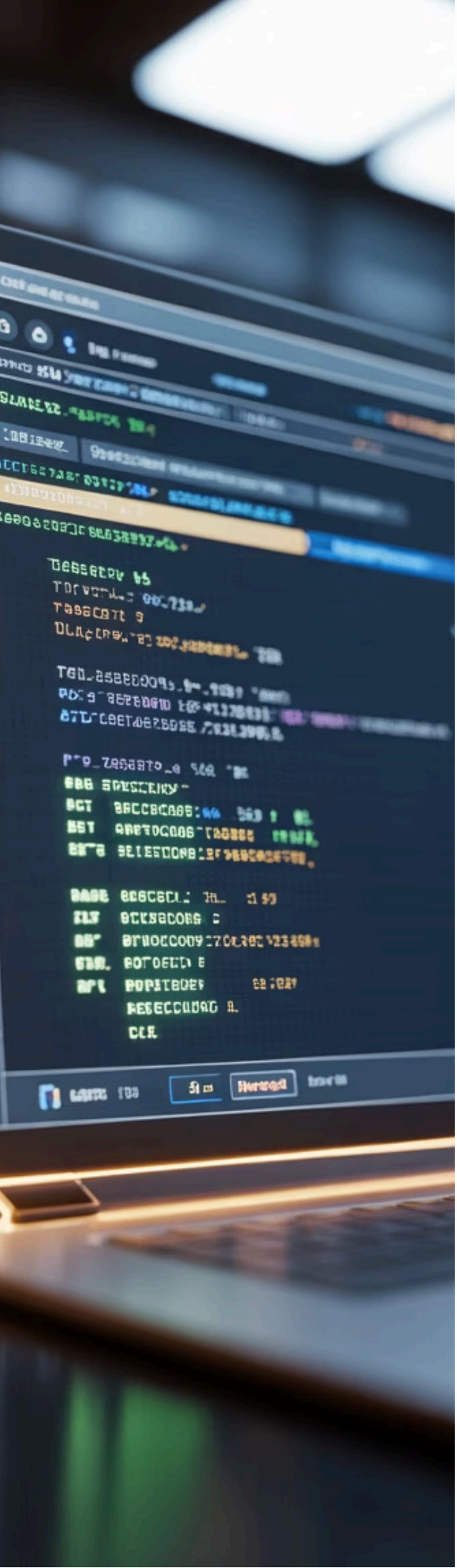
Autor: Josué Elías Tepet Zet

Fecha: Octubre 2025

Curso: Algoritmos

Institución: Universidad Mariano Gálvez de Guatemala

Docente: Miguel Alejandro Catalán López



Descripción técnica general del sistema

El Gestor de Notas Académicas es un sistema desarrollado en Python que permite registrar, analizar y organizar las calificaciones de cursos mediante un menú interactivo. Utiliza estructuras de datos como listas, pilas y colas, y aplica algoritmos clásicos de ordenamiento y búsqueda. El sistema opera exclusivamente en consola y no requiere librerías externas.

Lenguaje

Desarrollado completamente en Python sin dependencias externas

Interfaz

Sistema de consola con menú interactivo y validaciones

Estructuras

Implementa listas, pilas y colas para gestión de datos

Estructura general del código

El programa está dividido en funciones modulares, cada una cumpliendo una tarea específica:

01

menu()

Controla la navegación del sistema

02

registrar_curso()

Agrega cursos y notas

03

mostrar_cursos()

Lista todos los registros

04

calcular_promedio()

Calcula el promedio general

05

contar_aprobados_reprobados()

Analiza los resultados

Funciones de modificación

- actualizar_nota(): permite modificaciones controladas
- eliminar_curso(): elimina registros con confirmación

Funciones de ordenamiento y búsqueda

- ordenar_por_nota() y ordenar_por_nombre(): aplican ordenamientos
- buscar_curso_lineal() y buscar_curso_binaria(): realizan búsquedas

Funciones de estructuras avanzadas

- simular_cola_revision(): simula una cola de revisión de notas
- mostrar_historial(): muestra la pila de cambios recientes

Uso de estructuras de datos

El sistema implementa tres estructuras de datos fundamentales para la gestión eficiente de la información académica:

Lista principal (cursos)

Almacena los nombres de cursos y sus notas. Cada elemento es un diccionario con las claves 'nombre' y 'nota', permitiendo acceso directo y modificación flexible de los datos.

Pila (historial_cambios)

Guarda las modificaciones y eliminaciones en orden LIFO (Last In, First Out). Permite rastrear los cambios más recientes primero, útil para auditoría y deshacer operaciones.

Cola (cola_revision)

Simula el flujo de solicitudes en orden FIFO (First In, First Out). Representa el procesamiento secuencial de revisiones de notas, respetando el orden de llegada.

Algoritmos de ordenamiento y búsqueda

Justificación de los algoritmos implementados

Ordenamiento Burbuja

Se usa para ordenar los cursos según la nota, en orden descendente

Ordenamiento por Inserción

Se usa para ordenar los cursos alfabéticamente

Algoritmos de búsqueda

Búsqueda Lineal

Recorre toda la lista para encontrar coincidencias parciales. Ideal cuando no se requiere coincidencia exacta o cuando la lista no está ordenada.

Búsqueda Binaria

Encuentra coincidencias exactas en listas previamente ordenadas por nombre. Más eficiente que la búsqueda lineal con complejidad O(log n).

Justificación técnica: Se eligió el ordenamiento burbuja por su simplicidad didáctica para ordenar por notas, mientras que la inserción es más eficiente para listas pequeñas ordenadas alfabéticamente. La búsqueda binaria requiere ordenamiento previo pero ofrece mejor rendimiento en listas grandes.

Diagrama general del sistema

Inicio | Mostrar menú | Leer opción | Ejecutar función | Fin

El flujo principal del sistema sigue un patrón de menú iterativo donde el usuario selecciona opciones numéricas que ejecutan funciones específicas. Cada función realiza validaciones antes de modificar datos y retorna al menú principal hasta que el usuario decide salir.

Documentación breve de funciones

Módulos principales del sistema

Función	Descripción	Entrada	Salida
registrar_curso()	Agrega un nuevo curso y su nota	Nombre, nota	Confirmación de registro
mostrar_cursos()	Muestra todos los cursos registrados	N/A	Listado de cursos
calcular_promedio()	Calcula el promedio de notas	Lista de notas	Promedio numérico
ordenar_por_nota()	Ordena por nota (burbuja)	Lista de cursos	Lista ordenada
buscar_curso_binaria()	Busca curso exacto (binaria)	Nombre	Resultado encontrado

Funciones de gestión y control

actualizar_nota()

Permite actualizar la nota de un curso existente. Valida que la nueva nota esté en el rango 0-100 y registra el cambio en el historial antes de aplicarlo.

eliminar_curso()

Elimina un curso de la lista con confirmación del usuario. Guarda el registro eliminado en el historial de cambios para trazabilidad.

contar_aprobados_reprobados()

Cuenta cuántos cursos están aprobados (nota ≥60) y reprobados. Proporciona estadísticas rápidas del rendimiento académico.

simular_cola_revision()

Simula una cola de solicitudes de revisión de notas. Procesa las solicitudes en orden FIFO, demostrando el uso práctico de colas.

mostrar_historial()

Muestra los últimos cambios realizados en el sistema. Utiliza la pila para presentar los cambios más recientes primero (LIFO).

buscar_curso_lineal()

Realiza búsqueda lineal permitiendo coincidencias parciales. Útil cuando el usuario no recuerda el nombre exacto del curso.

Validaciones y control de errores

El sistema implementa múltiples capas de validación para garantizar la integridad de los datos:

Validación de entrada

- Nombres de curso no vacíos
- Notas en rango 0-100
- Valores numéricos válidos

Prevención de duplicados

- Verificación de cursos existentes
- Comparación case-insensitive
- Mensajes de advertencia claros

Confirmaciones

- Confirmación antes de eliminar
- Mensajes de éxito/error
- Manejo de excepciones

> Todas las funciones incluyen bloques try-except para capturar errores de conversión de tipos y validaciones específicas del dominio académico.

Conclusión técnica

El Gestor de Notas Académicas cumple con los requerimientos técnicos establecidos: **modularidad**, **validaciones**, aplicación de **algoritmos clásicos** y uso eficiente de estructuras lineales sin necesidad de librerías externas. Su diseño permite comprensión clara del flujo de datos y del control lógico.

13	3	4
Funciones modulares	**Estructuras de datos**	**Algoritmos clásicos**
Cada una con responsabilidad única y bien definida	Listas, pilas y colas implementadas correctamente	Ordenamiento y búsqueda con justificación técnica

Autor: Josué Elías Tepet Zet | **Universidad Mariano Gálvez de Guatemala** | Octubre 2025