Universidad: Mariano Gálvez

Materia: Algoritmos

Ingeniero: Miguel Catalán

Carrera: Ingeniería en sistemas


# Manual técnico del Gestor de notas académicas

Nombre del alumno: Obed Isaí

Apellidos del alumno: Sontay Delgado

Carnet: 7590-25-9847

## 1. Descripción técnica general del sistema

El Gestor Académico de Notas es una aplicación de consola escrita en Python que permite registrar cursos con sus notas, visualizar el listado de cursos, calcular promedio, buscar y eliminar cursos, mantener un historial (pila) de acciones y una cola de solicitudes de revisión. Además, incluye funciones para ordenar las notas mediante algoritmos clásicos (burbuja e inserción).

Objetivo: Practicar estructuras de datos básicas (listas, pilas y colas) y algoritmos de ordenamiento y búsqueda en un sistema funcional.

## 2. Estructura general del código

Archivo principal: gestor_notas.py
Estructura interna (resumen):
Variables:
cursos (lista de listas): almacena elementos [nombre, nota].
historial (lista): actuando como pila; se agregan strings como "Agregado Algebra".
cola (lista): actuando como cola FIFO; se agregan nombres de cursos pendientes.

- Funciones:

agregar_curso()
ver_cursos()
calcular_promedio()
buscar_curso() (búsqueda lineal)
eliminar_curso()
ver_historial() (muestra pila invertida)
revisar_cola() (pop(0) para simular atención FIFO)
ordenar_burbuja() (ordenamiento por nota ascendente)
ordenar_insercion() (ordenamiento por nota ascendente)
Bucle principal while True: con menú de usuario

## 3. Explicación del uso de listas, pilas y colas

**Lista (cursos)**: Se usa una lista de listas donde cada elemento es [nombre_del_curso, nota]. Permite almacenamiento dinámico y acceso por índice.
**Pila (historial)**: Implementada con una simple lista donde se append() cada acción. Para mostrar en orden LIFO se itera con reversed(historial) o se usan pop() en otro escenario.
**Cola (cola)**: Implementada con una lista. Para simular FIFO se utiliza cola.pop(0) al procesar la primera solicitud. (Nota: para producción sería mejor collections.deque por eficiencia).

## 4. Justificación de los algoritmos de ordenamiento implementados

**Burbuja (bubble sort)**: algoritmo sencillo de comparación e intercambio, fácil de implementar y entender; apropiado para un proyecto educativo donde la claridad importa. Complejidad $O(n^2)$, eficiente solo para listas pequeñas.
**Inserción (insertion sort)**: algoritmo intuitivo que construye la lista ordenada elemento a elemento; también $O(n^2)$ en promedio, pero suele ser eficiente para listas parcialmente ordenadas. Se implementa para comparar dos enfoques básicos.
Se eligieron estos algoritmos por su valor didáctico: muestran conceptos de comparación, desplazamiento e intercambio sin dependencias externas.

### 5. Documentación breve de cada función / módulo

agregar_curso(): pide nombre y nota; valida entrada; agrega [nombre,nota] a cursos,añade entrada a historial y cola.
Errores manejados: nombre vacío, nota no numérica, nota fuera de rango [0,100].
ver_cursos(): lista todos los cursos con su índice y nota. Si no hay cursos muestra mensaje.
calcular_promedio(): suma notas y divide entre cantidad; muestra promedio con 1 decimal.
buscar_curso(): petición por nombre; comparación case-insensitive; imprime nota si lo encuentra.
eliminar_curso(): busca por nombre y elimina el primer match; agrega acción al historial.
ver_historial(): muestra historial en orden de último a primero.
revisar_cola(): procesa la primera entrada de cola con pop(0).
ordenar_burbuja(): ordena cursos por nota ascendente intercambiando elementos.
ordenar_insercion(): ordena cursos por nota ascendente moviendo elementos hacia la izquierda.

### 6. Diagrama general del sistema / Pseudocódigo principal

```
INICIO
 lista cursos = []
ciclo while True
mostrar menú
leer opción
SI opción == 1:  registrar_curso_nota()
SI opción == 2: mostrar_curso_nota()
SI opción == 3: calcular_promedio()
SI opción == 4: cursos_aprovados_reprovados()
SI opción == 5: buscar_curso_nombre()
SI opción == 6: actualizar_nota_curso()
SI opción == 7: eliminar_curso()
SI opción == 8: ordenamiento_burbuja()
SI opción == 9: ordenamiento_insercion()
SI opción == 10: busqueda_curso()
SI opción == 11: simular_cola_revision()
SI opción == 12: mostrar_historial_cambios()
SI opción == 13: SALIR FIN
```