Manual de Usuario – 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

Índice

Información del documento

Versión

1.0

Última actualización

Octubre 2025

Páginas totales

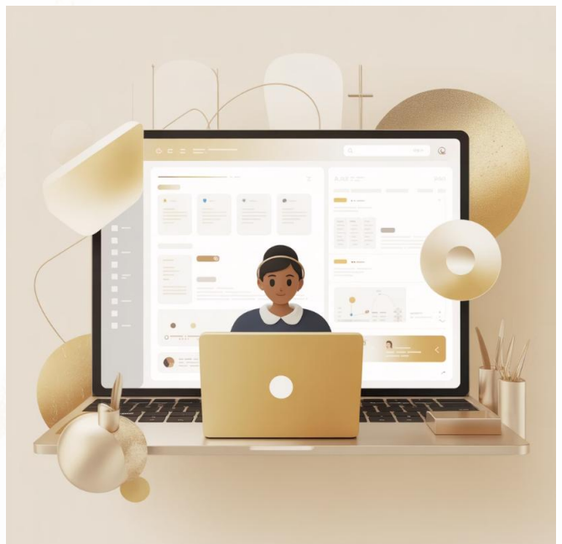
16

Formato

Manual técnico académico

Propósito del sistema

El sistema permite a los estudiantes registrar, consultar y analizar sus calificaciones académicas de manera simple y funcional, directamente desde la consola.



Beneficios Educativos

Este gestor fomenta la autoevaluación y la identificación temprana de áreas de mejora. Al visualizar su progreso académico, los estudiantes pueden ajustar sus estrategias de estudio y desarrollar una mayor autonomía en su proceso de aprendizaje.

Objetivos de Aprendizaje

Más allá de la gestión de notas, este proyecto tiene como objetivo principal reforzar el pensamiento lógico, la organización estructurada de datos y la resolución de problemas complejos. Sirve como una aplicación práctica de los algoritmos y estructuras de datos enseñados en el curso, permitiendo a los estudiantes manipular información de forma eficiente.

Desarrollo de Habilidades de Programación

Mediante la interacción con el sistema en consola, los estudiantes practican la implementación de interfaces de usuario básicas, el manejo de entrada/salida (I/O), la modularización de código, la gestión de errores y la depuración. Es una herramienta clave para solidificar los conceptos fundamentales de la programación y la ingeniería de software.

Metodología y Conceptos Practicados

La construcción y el uso de este sistema se alinea con una metodología de aprendizaje "aprender haciendo". Los conceptos clave practicados incluyen:

- Variables
- Estructuras de control (condicionales y bucles)
- Arreglos o listas
- Funciones/métodos
- Aplicación de algoritmos de búsqueda y ordenamiento

Todo ello en un entorno de desarrollo de consola que simula escenarios reales de aplicación de software.

Requisitos para ejecutar el programa

Requisitos de software

Python

Versión mínima: Python 3.7 o superior

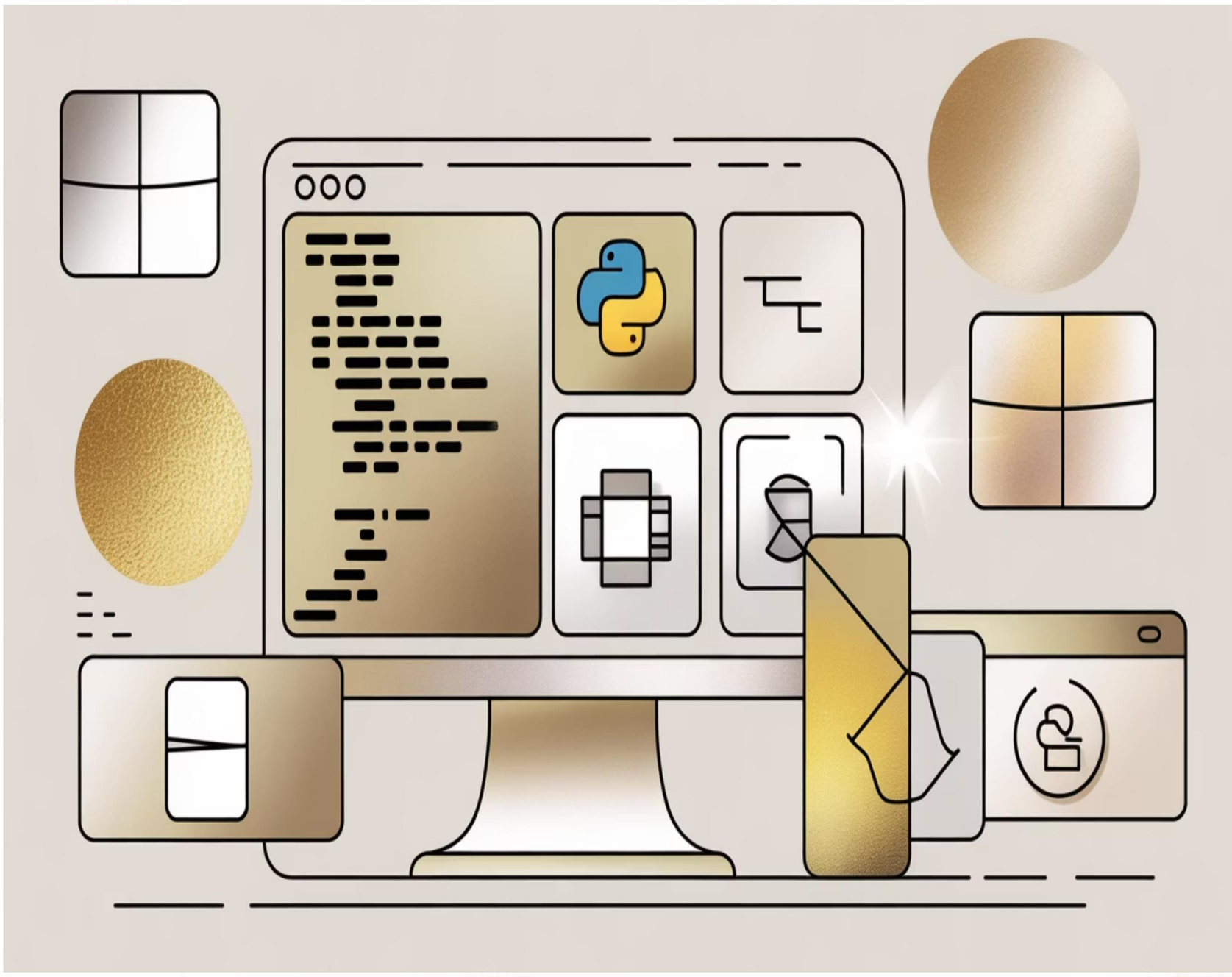
Versión recomendada: Python 3.9 o superior

Librerías estándar: El sistema utiliza únicamente librerías incluidas en Python (no requiere instalaciones adicionales)

Intérprete de comandos

Windows: Command Prompt (cmd) o PowerShell

macOS/Linux: Terminal o cualquier shell compatible



Requisitos de hardware

Memoria RAM

Mínimo: 512 MB de RAM disponible

Recomendado: 1 GB de RAM para un rendimiento óptimo

Almacenamiento

Espacio requerido: 50 MB de espacio libre en disco

Espacio recomendado: 100 MB para archivos temporales y logs

Procesador

Mínimo: Procesador de 1 GHz

Compatible: Cualquier arquitectura x86, x64, ARM

Sistemas operativos compatibles

Windows: Windows 7, 8, 10, 11 (32-bit y 64-bit)

macOS: macOS 10.12 Sierra o superior

Linux: Distribuciones principales (Ubuntu, Debian, CentOS, Fedora)

Otros: Cualquier sistema que soporte Python 3.7+

Verificación de instalación

Para verificar que tu sistema cumple con los requisitos:

```
# Verificar versión de Pythonpython --version# o en algunos sistemaspython3 --version# Verificar que Python funciona correctamentepython -c "print('Python está funcionando correctamente')"
```

Dependencias del sistema

No se requieren dependencias externas. El sistema está diseñado para funcionar únicamente con las librerías estándar de Python, lo que garantiza:

- Instalación simple sin configuraciones complejas
- Compatibilidad amplia entre diferentes sistemas
- Menor riesgo de conflictos de dependencias

Ejemplos de uso

01

Registrar nuevos cursos

Registro inicial de dos cursos con sus respectivas notas.

```
> Registrar cursoIngrese el nombre del curso: MatemáticasIngrese la nota obtenida: 85Curso registrado con éxito.> Registrar cursoIngrese el nombre del curso: ProgramaciónIngrese la nota obtenida: 95Curso registrado con éxito.
```

02

Listar todos los cursos registrados

Visualiza rápidamente todos los cursos que has registrado.

```
> Listar cursosLista de cursos registrados:1. Matemáticas - Nota: 852. Programación - Nota: 95
```

03

Calcular el promedio general de notas

Obtén el promedio ponderado de todas tus calificaciones.

```
> Calcular promedioPromedio general: 90.0
```

04

Actualizar la nota de un curso existente

Corrige o actualiza la calificación de un curso específico.

```
> Actualizar cursoIngrese el nombre del curso a actualizar: MatemáticasIngrese la nueva nota: 90Nota de Matemáticas actualizada con éxito a 90.
```

05

Calcular el promedio después de una actualización

Verifica cómo la actualización impacta tu promedio general.

```
> Calcular promedioPromedio general: 92.5
```

06

Ordenar cursos por nota (descendente)

Organiza tus cursos para ver rápidamente tus mejores y peores desempeños.

```
> Ordenar cursos por notaCursos ordenados por nota:1. Programación - Nota: 952. Matemáticas - Nota: 90
```

Ejemplos avanzados y manejo de errores

7. Buscar un curso específico

Encuentra la información detallada de un curso rápidamente.

```
> Buscar cursoIngrese el nombre del curso a buscar: ProgramaciónCurso encontrado: Programación - Nota: 95.
```

8. Eliminar un curso

Remueve un curso de tu registro, por ejemplo, si lo has retirado.

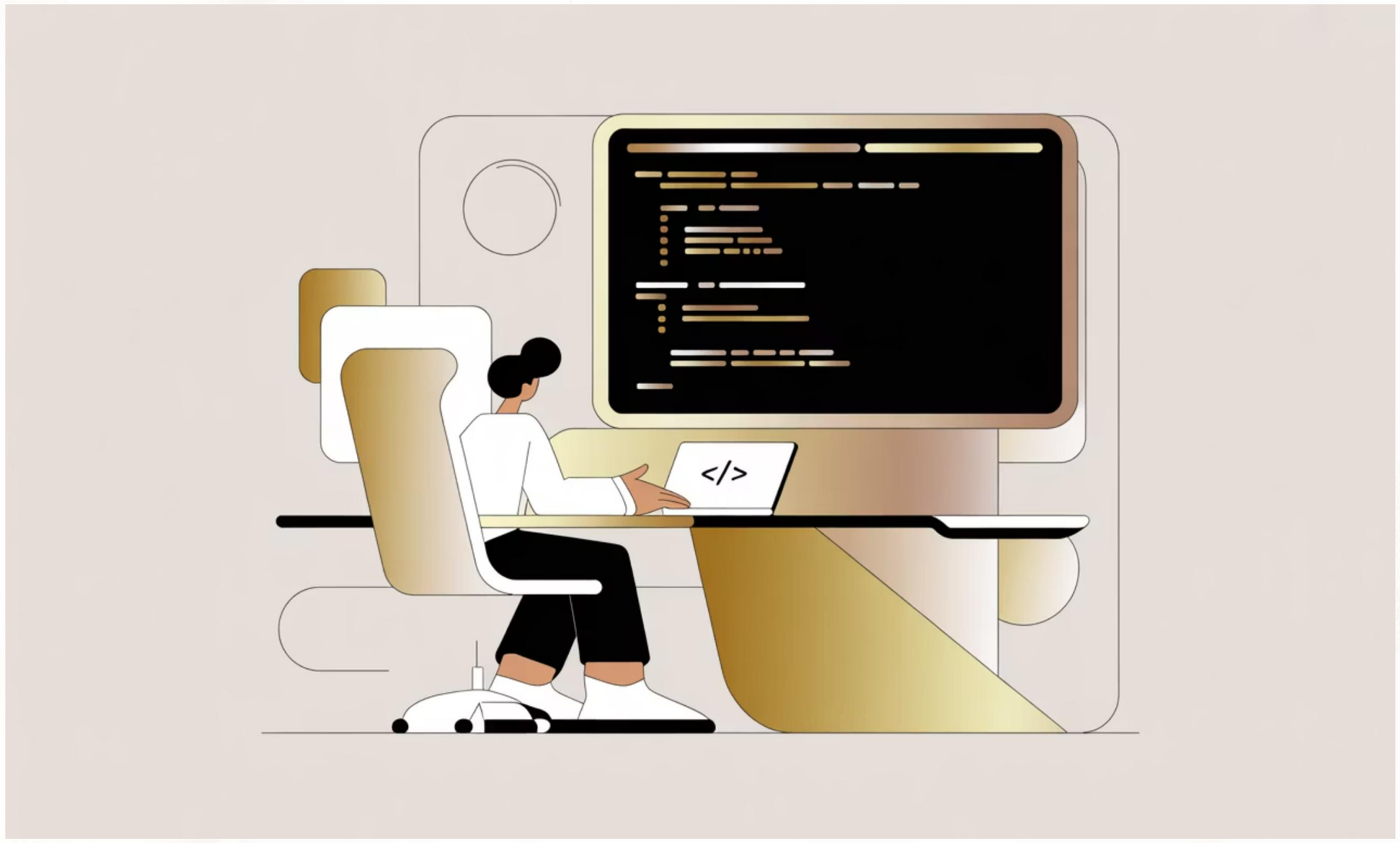
```
> Eliminar cursoIngrese el nombre del curso a eliminar: Matemáticas¿Está seguro de eliminar el curso 'Matemáticas'? (s/n): sCurso Matemáticas eliminado con éxito.
```

9. Verificar lista después de eliminación

Confirma que el curso ha sido removido de la lista.

```
> Listar cursosLista de cursos registrados:1. Programación - Nota: 95
```

Manejo de errores del sistema



10. Manejo de errores: Nota inválida

```
> Registrar cursoIngrese el nombre del curso: FísicaIngrese la nota obtenida: abcError: La nota debe ser un valor numérico entre 0 y 100.
```

11. Manejo de errores: Curso no encontrado

```
> Actualizar cursoIngrese el nombre del curso a actualizar: QuímicaError: Curso 'Química' no encontrado.
```

12. Manejo de errores: Curso duplicado

```
> Registrar cursoIngrese el nombre del curso: ProgramaciónError: El curso 'Programación' ya existe. Use la opción actualizar.
```

Funcionalidades principales del sistema

Gestión básica

Funciones esenciales para el mantenimiento y la consulta de la información académica, presentadas de forma concisa.

Registrar nuevo curso

Permite añadir un nuevo curso al sistema con un nombre único y una nota entre 0 y 100. Es fundamental para construir el historial académico, mostrando un mensaje de confirmación o error si el curso ya existe o la nota es inválida.

Mostrar todos los cursos y notas

Presenta una lista completa y formateada de todos los cursos registrados y sus notas. Ideal para revisar rápidamente el progreso académico o verificar la entrada de datos.

Calcular promedio general

Calcula el promedio aritmético de todas las notas, ofreciendo una visión general del rendimiento académico. Retorna un valor numérico o un mensaje si no hay cursos.

Contar cursos aprobados y reprobados

Determina la cantidad de cursos por encima o por debajo de un umbral de aprobación predefinido. Proporciona una estadística simple de éxito académico para informes o autoevaluación.

Actualizar nota de un curso

Permite modificar la calificación de un curso existente, útil para correcciones o ajustes. Requiere el nombre del curso y la nueva nota, con mensajes de confirmación o error.

Eliminar un curso

Remueve permanentemente un curso y su nota asociada del registro. Es una operación irreversible que requiere el nombre del curso, mostrando confirmación o error si no se encuentra.

Funcionalidades avanzadas del sistema

Algoritmos de búsqueda

Buscar curso por nombre (búsqueda lineal)

Recorre secuencialmente toda la lista hasta encontrar el curso solicitado.

- **Complejidad:** O(n)
- **Uso:** Cuando la lista no está ordenada
- **Ventaja:** Simple de implementar

Buscar curso por nombre (búsqueda binaria)

Búsqueda eficiente que requiere que la lista esté previamente ordenada.

- **Complejidad:** O(log n)
- **Uso:** En listas ordenadas alfabéticamente
- **Ventaja:** Muy eficiente para listas grandes

Algoritmos de ordenamiento

Ordenar cursos por nota (método burbuja)

Organiza los cursos de mayor a menor nota usando comparaciones adyacentes.

- **Complejidad:** $O(n^2)$
- **Uso:** Para visualizar rendimiento académico
- **Resultado:** Lista ordenada descendentemente

Ordenar cursos por nombre (método inserción)

Ordena alfabéticamente los nombres de los cursos.

- **Complejidad:** $O(n^2)$
- **Uso:** Para organización alfabética
- **Resultado:** Lista ordenada ascendentemente

Estructuras de datos y arquitectura del sistema

Estructuras de datos principales

- **Lista principal:** Almacena objetos de tipo curso con atributos nombre y nota.
- **Pila de historial:** Registra cambios realizados para funcionalidad de deshacer. Utiliza una pila LIFO para registrar las últimas modificaciones realizadas.
  - **Operaciones:** Push, pop, ver tope
  - **Uso educativo:** Demuestra el concepto de pila
- **Cola de solicitudes:** Simula un sistema de revisión de notas pendientes. Implementa una cola FIFO para gestionar solicitudes de revisión de notas.
  - **Operaciones:** Encolar, desencolar, ver siguiente
  - **Uso educativo:** Demuestra el concepto de cola

Arquitectura del sistema

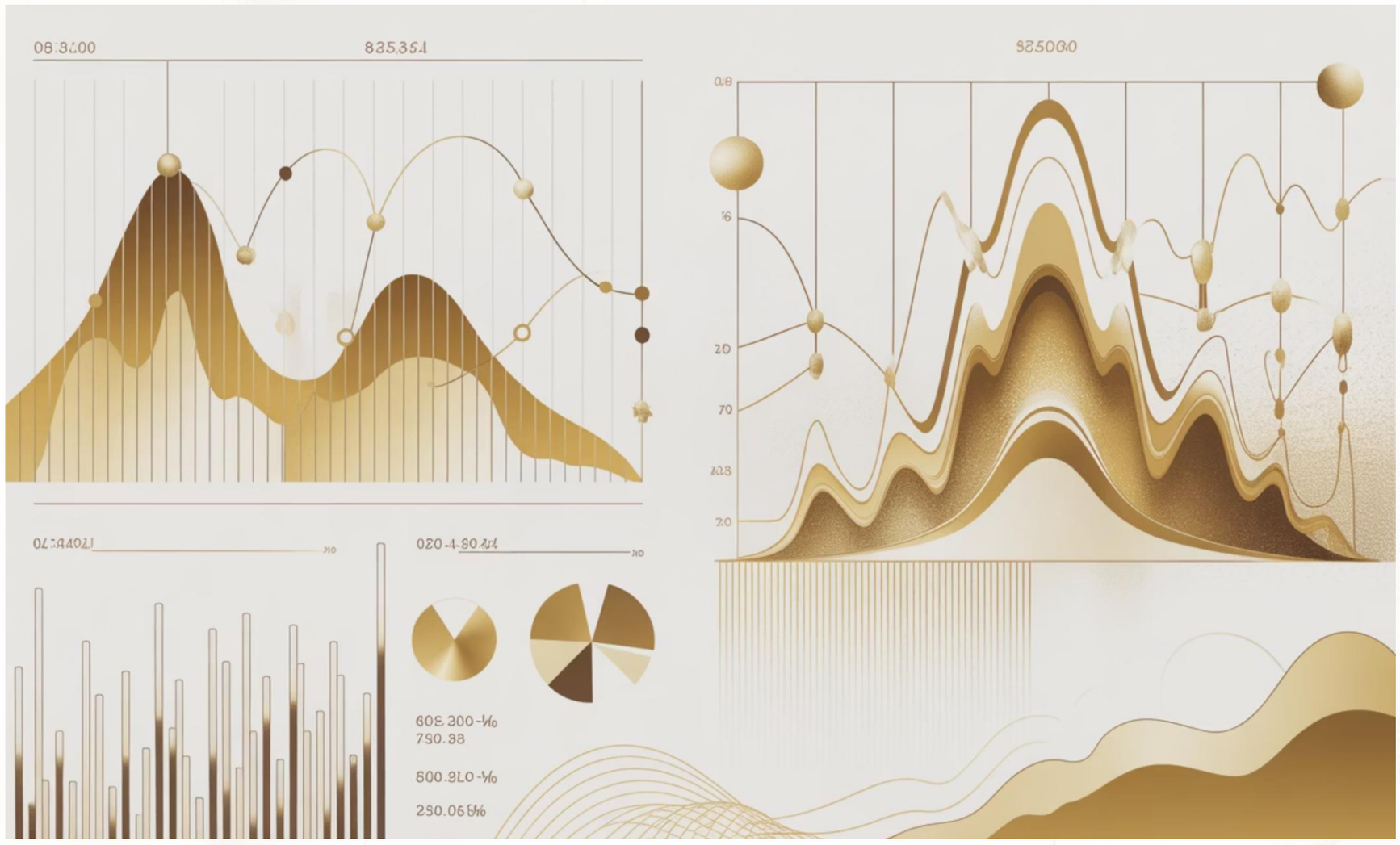
El sistema está construido siguiendo principios de programación estructurada y utiliza un enfoque modular para organizar las funcionalidades.

Flujo de ejecución

1. **Inicialización:** Inicialización de estructuras de datos vacías
2. **Presentación de Menú:** Presentación del menú principal
3. **Captura y Validación:** Captura y validación de entrada del usuario
4. **Ejecución:** Ejecución de la función correspondiente
5. **Actualización:** Actualización de estructuras de datos
6. **Retorno:** Retorno al menú principal

Consideraciones de rendimiento y limitaciones

El análisis de rendimiento es fundamental para entender la eficiencia del sistema y tomar decisiones informadas sobre qué algoritmos utilizar según el contexto.



Complejidad temporal de operaciones

Operación	Complejidad	Descripción
Inserción	O(1)	Agregar al final de la lista
Búsqueda lineal	O(n)	Recorre toda la lista
Búsqueda binaria	O(log n)	Requiere lista ordenada
Ordenamiento burbuja	$O(n^2)$	Comparaciones anidadas
Ordenamiento inserción	$O(n^2)$	En el peor caso

Análisis de eficiencia

Operaciones más eficientes:

- Inserción de nuevos cursos: Tiempo constante
- Acceso directo por índice: Inmediato

Operaciones que requieren optimización:

- Búsquedas en listas grandes: Considerar ordenamiento previo
- Ordenamiento de grandes volúmenes: Evaluar algoritmos más eficientes

Limitaciones actuales del sistema

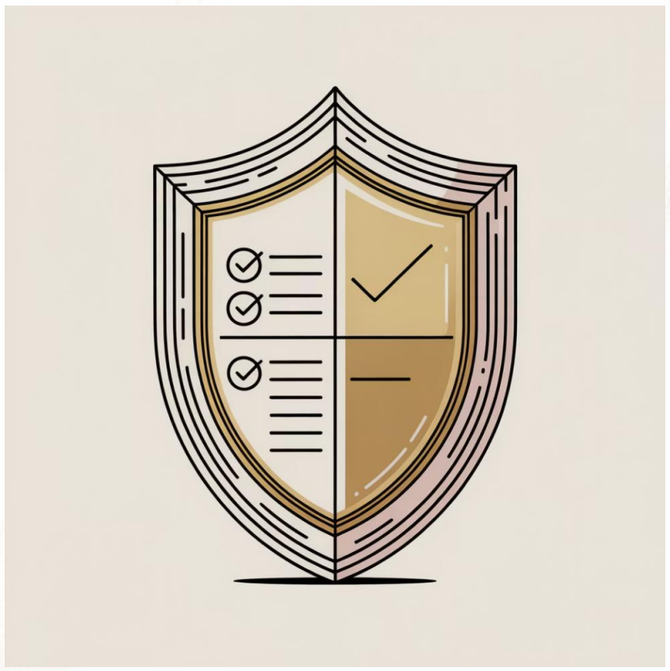
Limitaciones técnicas

- **Capacidad limitada por memoria RAM disponible:** El sistema almacena todos los datos en memoria principal
- **Sin persistencia de datos entre sesiones:** Los datos se pierden al cerrar la aplicación
- **Interfaz únicamente por consola:** No cuenta con interfaz gráfica de usuario
- **Sin soporte para múltiples usuarios:** Diseñado para uso individual

Recomendaciones para versiones futuras

- Implementar persistencia en archivos o base de datos
- Desarrollar interfaz gráfica de usuario
- Agregar soporte para múltiples usuarios simultáneos
- Optimizar algoritmos para grandes volúmenes de datos

Validaciones y seguridad del sistema



El sistema implementa **múltiples validaciones** para garantizar la integridad de los datos y la seguridad, protegiendo contra errores comunes y usos indebidos.

Validaciones de datos y flujo

Validación de Notas

Asegura que todas las calificaciones se mantengan dentro de un rango aceptable (ej. 0-100), mostrando un mensaje de error si el valor está fuera de rango.

Prevención de Duplicados

Impide la creación de registros idénticos (ej. cursos con el mismo nombre) para evitar confusión o inconsistencia en los datos.

Validación de Entrada

Verifica que los datos ingresados sean del tipo y formato correctos (ej. nombre no vacío, nota numérica), informando al usuario sobre el formato esperado en caso de error.

Confirmación de Eliminación

Solicita verificación antes de realizar acciones irreversibles (ej. eliminar un curso) para evitar la pérdida accidental de datos.

Controles de Seguridad Adicionales

Sanitización de Entrada (Input Sanitization)

Procesa y limpia las entradas de texto para eliminar caracteres o secuencias que puedan ser maliciosas, protegiendo contra ataques de inyección (ej. XSS).

Manejo de Errores Controlado

Presenta mensajes de error genéricos al usuario final para evitar revelar detalles internos del sistema, mientras registra los detalles técnicos para los administradores.

Restricción de Acceso (Autorización)

Asegura que solo los usuarios autorizados puedan realizar ciertas operaciones, verificando las credenciales antes de ejecutar acciones críticas.

Registro de Actividad (Auditoría)

Mantiene un registro de las acciones importantes realizadas en el sistema (quién, qué, cuándo), permitiendo rastrear cambios y cumplir con requisitos de auditoría.

Preguntas frecuentes

¿Qué pasa si ingreso una nota fuera del rango?

El sistema mostrará un mensaje de error claro, como "Error: La nota debe estar entre 0 y 100. Por favor, introduce un valor válido.", y no registrará el dato. Esto es parte de las validaciones de datos para asegurar la integridad.

¿Se guardan los datos al cerrar el programa?

No, los datos existen únicamente durante la ejecución de la aplicación, ya que se manejan en memoria utilizando estructuras como listas, pilas y colas. Al cerrar el programa, toda la información ingresada se perderá, a menos que se implemente una solución de persistencia externa (como archivos o bases de datos).

¿Cómo detengo el programa?

Para salir del programa de forma segura, selecciona la opción número 13, que corresponde a "Salir" en el menú principal. Esto asegura que el programa finalice correctamente.

¿Por qué mi entrada parece correcta pero el sistema la rechaza?

A menudo, esto se debe a las validaciones de entrada. El sistema verifica no solo el rango de los datos (como en las notas), sino también el formato y si ya existen duplicados. Por ejemplo, si intentas registrar un curso con un nombre que ya existe, o si ingresas texto donde se espera un número, el sistema lo rechazará para mantener la consistencia de los datos. Revisa los mensajes de error; son muy específicos.

¿Cómo evita el sistema los registros duplicados?

El sistema tiene una función de prevención de duplicados que impide la creación de registros idénticos, como cursos con el mismo nombre o estudiantes con la misma identificación, para mantener la unicidad y consistencia de los datos.

¿Qué tipo de información se considera "maliciosa" en la entrada de datos?

La "sanitización de entrada" se encarga de eliminar caracteres o secuencias de texto que podrían ser utilizados para ataques de inyección (como XSS). Esto incluye scripts, código HTML o cualquier dato inesperado en campos de texto que podrían comprometer la seguridad o la integridad del sistema.

¿Qué sucede si un usuario no autorizado intenta realizar una acción?

El sistema implementa restricciones de acceso (autorización) que verifican las credenciales del usuario. Si un usuario no tiene los permisos necesarios para realizar una operación específica, el sistema le negará el acceso y, en algunos casos, registrará el intento.

¿Puedo ver quién hizo un cambio específico en el sistema?

Sí, el sistema mantiene un registro de actividad (auditoría) que graba las acciones importantes. Esto incluye quién realizó una acción, qué acción fue, y cuándo se llevó a cabo, facilitando el seguimiento de cambios y la conformidad con auditorías.

¿Qué tipo de mensajes de error me dará el sistema?

El sistema mostrará mensajes de error genéricos para los usuarios finales para no revelar información sensible del sistema. Sin embargo, se registrarán detalles técnicos de los errores para que los administradores puedan diagnosticarlos y resolverlos.

¿Qué pasa si intento eliminar algo importante por accidente?

El sistema requiere una confirmación antes de realizar acciones irreversibles, como la eliminación de un curso o un registro de estudiante. Esto ayuda a prevenir la pérdida accidental de datos y asegura que las decisiones sean intencionales.

¿Existen limitaciones en el formato de los datos que puedo introducir?

Sí, la validación de entrada verifica que los datos cumplan con el tipo y formato correctos. Por ejemplo, si se espera un número, no se aceptará texto. El sistema proporcionará mensajes de error específicos indicando el formato esperado para guiar al usuario.

¿Cómo puedo estar seguro de que mis datos están protegidos contra usos indebidos?

El sistema implementa una combinación de sanitización de entrada, restricción de acceso, manejo controlado de errores y registro de actividad para proteger contra usos indebidos, garantizar la integridad de los datos y mantener la seguridad general del sistema.

Algoritmos implementados

El sistema utiliza algoritmos clásicos de programación para demostrar conceptos fundamentales de estructuras de datos y eficiencia:

Algoritmos de ordenamiento

Estos algoritmos reorganizan los elementos de una lista en una secuencia específica (ascendente o descendente).

Método Burbuja (Bubble Sort)

Explicación Técnica: Es un algoritmo de ordenamiento simple que funciona comparando y cambiando de lugar pares de elementos adyacentes si están en el orden incorrecto. Las pasadas se repiten hasta que la lista está ordenada.

Complejidad Temporal:

- **Mejor Caso (ya ordenado):** O(n)
- **Promedio y Peor Caso:** O(n^2)

Casos de Uso Específicos: Principalmente educativo, o para conjuntos de datos muy pequeños.

Ventajas: Muy fácil de entender e implementar. **Desventajas:** Extremadamente ineficiente para grandes conjuntos de datos.

Pseudocódigo:

```
PROCEDIMIENTO bubbleSort(lista):  n = longitud de lista  repetir    intercambiado = FALSO    PARA i DESDE 0 HASTA n-2:      SI lista[i] > lista[i+1]:        intercambiar lista[i] y lista[i+1]        intercambiado = VERDADERO  MIENTRAS intercambiado ES VERDADEROFIN PROCEDIMIENTO
```

Método de Inserción (Insertion Sort)

Explicación Técnica: Construye el array ordenado un elemento a la vez. Itera sobre los elementos de entrada y para cada uno, lo inserta en su posición correcta dentro de la sublista ya ordenada.

Complejidad Temporal:

- **Mejor Caso (ya ordenada):** O(n)
- **Promedio y Peor Caso:** O(n^2)

Casos de Uso Específicos: Conjuntos de datos pequeños, datos ya sustancialmente ordenados, o cuando los elementos se ordenan incrementalmente.

Ventajas: Simple de implementar, eficiente para pequeños conjuntos de datos, estable y de bajo impacto en la memoria (in-place). **Desventajas:** Ineficiente para grandes conjuntos de datos.

Pseudocódigo:

```
PROCEDIMIENTO insertionSort(lista):  n = longitud de lista  PARA i DESDE 1 HASTA n-1:    clave = lista[i]    j = i - 1    MIENTRAS j >= 0 Y clave < lista[j]:      lista[j + 1] = lista[j]      j = j - 1    lista[j + 1] = claveFIN PROCEDIMIENTO
```

Algoritmos de búsqueda

Búsqueda Lineal

Explicación Técnica: Recorre secuencialmente cada elemento de la lista hasta encontrar el objetivo o llegar al final.

Complejidad: O(n) en el peor caso

Ventajas: Simple, funciona en listas no ordenadas

Desventajas: Lento para listas grandes

Búsqueda Binaria

Explicación Técnica: Divide repetidamente la lista ordenada por la mitad, comparando el elemento central con el objetivo.

Complejidad: O(log n)

Ventajas: Muy eficiente para listas grandes

Desventajas: Requiere lista previamente ordenada

Estructuras de datos implementadas

- **Listas (Arrays Dinámicos):** Almacenamiento principal de cursos
- **Pila (Stack - LIFO):** Historial de cambios recientes
- **Cola (Queue - FIFO):** Simulación de solicitudes de revisión

Conclusión del usuario

El Gestor de Notas Académicas permite una experiencia práctica e intuitiva para el manejo de calificaciones personales.

Su interfaz por consola es clara y accesible, y cumple con los objetivos de enseñanza del curso de programación.

Características destacadas:

- Interfaz de consola simple y directa
- Validaciones robustas de datos
- Implementación de algoritmos clásicos
- Gestión completa del ciclo de vida de las notas
- Herramienta educativa efectiva

Reflexión final

La creación de este gestor ha sido una experiencia de aprendizaje transformadora, cultivando habilidades críticas de pensamiento computacional y apreciación por la arquitectura de software.

Anexos y Referencias

Anexo A: Códigos de error del sistema

Código	Descripción	Solución
E001	Nota fuera de rango (0-100)	Ingresar valor numérico válido
E002	Curso duplicado	Usar nombre diferente o actualizar existente
E003	Curso no encontrado	Verificar nombre exacto del curso
E004	Lista vacía	Registrar al menos un curso primero
E005	Entrada no numérica	Ingresar solo números para las notas

Anexo B: Comandos de terminal útiles

Verificar instalación de Python

```
python --versionpython3 --version
```

Ejecutar el programa

```
# Windowspython gestor_notas.py# macOS/Linuxpython3 gestor_notas.py
```

Navegar en terminal

```
# Listar archivosdir (Windows) / ls (macOS/Linux)# Cambiar directoriocd nombre_carpeta# Directorio actualpwd
```

Referencias Bibliográficas

Se consultaron diversas fuentes en línea y documentación oficial de Python para la implementación de estructuras de datos, manejo de excepciones y buenas prácticas de programación.

Fin del manual