Universidad Continental

Unidad 03

Universidad Continental

Apellidos y nombres: Michue Izquierdo Jhojandy Azariel

Carrera: Ing. sistemas e informática

Curso: Estructura de datos

Docente: Davy Dario Veli Rojas

Estructura del Informe

CAPITULO 1: Análisis del Problema

Descripción del problema

*(*En un salón con 20–30 alumnos, el profesor necesita llevar las notas de todos. Hacerlo en papel o en Excel puede ser cansado, se pueden equivocar los promedios o perder la información.

Entonces creamos un sistema sencillo que permita:

- Registrar alumnos con 3 notas.
- Calcular el promedio automáticamente.
- Mostrar la lista de todos los alumnos.
- Más adelante: eliminar, guardar y cargar.

La idea es que el profesor pueda revisar las notas rápido y sin errores..*)*

1. **Requerimientos del sistema**
   - **Funcionales**
     - Registrar un alumno con su nombre y sus 3 notas.
     - Calcular el promedio automático.
     - Guardar a todos los alumnos en una lista.
     - Mostrar la lista completa con su promedio.
     - Eliminar un alumno por su ID.
     - Guardar y cargar los datos.

   - **No funcionales**
     - Debe ser fácil de usar, con un menú claro.
     - Debe ser simple, sin complicación.
     - Debe dar mensajes claros si ingresas algo mal.
     - Debe funcionar en Google Colab sin instalar nada.

2. **Estructuras de datos propuestas**

Una clase Alumno -> para guardar los datos de cada estudiante.

Una lista llamada alumnos -> para guardar a todos los estudiantes registrados.

Una variable siguiente_id -> para que cada alumno tenga un ID único.

Ejemplo: ID / Nombre / Nota1 / Nota2 / Nota3 / Promedio

- **Justificación de la elección**

*(*La clase Alumno deja todo ordenado (nombre, notas, promedio).

La lista alumnos nos deja guardar entre 20 y 30 estudiantes sin límite fijo.

El ID automático evita confusiones si dos alumnos se llaman igual.

En resumen: son simples, claras y suficientes para manejar notas de un salón.*)*

Capítulo 2: Diseño de la Solución

1. **Descripción de estructuras de datos y operaciones:**

Opción 1 - Registrar alumno

- Pide nombre y tres notas.
- Crea un objeto Alumno con un ID automático.
- Calcula el promedio.
- Lo agrega a la lista alumnos.

Opción 2 - Listar alumnos

- Recorre la lista alumnos.
- Muestra por pantalla: ID, nombre, notas y promedio.

Opción 3 - Eliminar alumno

- Pide el ID del alumno.
- Busca en la lista si hay un alumno con ese ID.
- Si lo encuentra, lo elimina de la lista.
- Si no existe, muestra un mensaje de error.

Opción 4 - Guardar datos

- Convierte cada alumno en un diccionario (usando a_dict() de la clase alumno).
- Arma una estructura con:

        - Siguiente_id
        - la lista de alumnos en diccionarios
- Escribe todo en el archivo alumnos.json

2. **Algoritmos principales:**
   - ***Pseudocódigo para registrar alumno.***

```
INICIO registrar_alumno
  mostrar "Registrar alumno"
  leer nombre
  leer nota1
  leer nota2
  leer nota3

  calcular promedio = (nota1 + nota2 + nota3) / 3

  crear alumno con:
    id = siguiente_id
    nombre, nota1, nota2, nota3, promedio

  agregar alumno a la lista "alumnos"

  aumentar siguiente_id en 1

  mostrar "Alumno registrado correctamente"
FIN
```

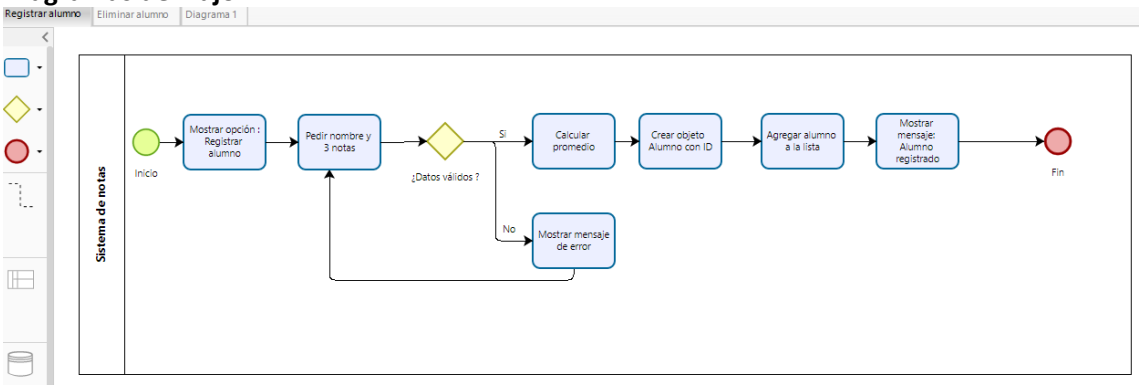
   - ***Pseudocódigo para eliminar alumno por ID.***

```
INICIO eliminar_alumno
  si la lista "alumnos" está vacía
    mostrar "No hay alumnos para eliminar"
    FIN

  leer id_a_eliminar

  encontrado = FALSO

  para cada alumno en la lista "alumnos"
    si alumno.id == id_a_eliminar
      eliminar ese alumno de la lista
      mostrar "Alumno eliminado"
      encontrado = VERDADERO
      salir del ciclo

  si encontrado es FALSO
    mostrar "No existe un alumno con ese ID"
FIN
```

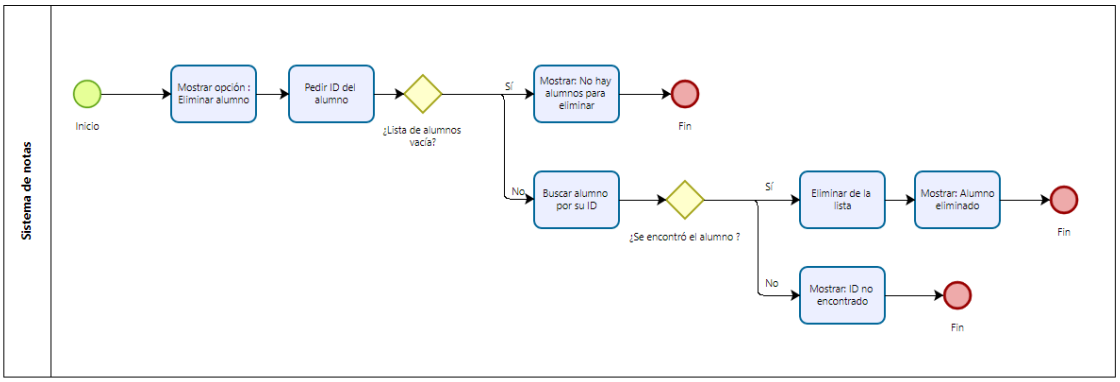
   - **Pseudocódigo para guardar datos**

```
INICIO guardar_datos
  crear lista_vacia llamada datos_alumnos

  para cada alumno en la lista "alumnos"
    convertir alumno a diccionario
    agregar el diccionario a datos_alumnos

  crear estructura a_guardar con:
    siguiente_id
    lista de datos_alumnos

  abrir archivo "alumnos.json" en modo escritura
  escribir a_guardar en formato JSON
  cerrar archivo

  mostrar "Datos guardados correctamente"
FIN
```

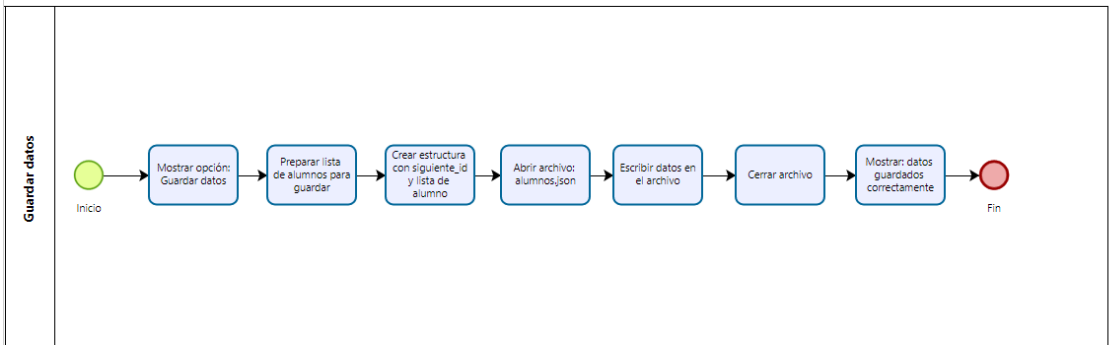
3. Diagramas de Flujo



Este diagrama muestra los pasos para ingresar los datos de un alumno, validar las notas, calcular el promedio y registrarlo.



Este diagrama representa el proceso para pedir el ID del alumno, verificar si existe y eliminarlo de la lista.



Muestra el proceso donde el sistema arma la estructura de datos y la guarda en el archivo JSON.

4. **Justificación del diseño:**
   (El sistema está hecho de forma sencilla para que cualquiera lo pueda usar sin complicarse.
   Cada opción del menú hace una cosa: registrar, listar, eliminar o guardar, así todo queda ordenado y fácil de entender.

   Usamos una clase `Alumno` porque así toda la información del estudiante queda junta y no regada por todos lados.
   Guardamos los alumnos en una lista porque es rápida, funciona perfecto para un salón de 20 a 30 estudiantes y permite agregar o borrar sin problemas.

   El menú es corto y claro, lo que ayuda a no perderse.
   También se valida que las notas sean números para evitar errores.

   Guardamos los datos en un archivo JSON porque es simple, práctico y fácil de usar si queremos recuperar la información después.

   En resumen: el diseño es directo, fácil de usar, fácil de entender y cumple bien con lo que necesitamos para manejar las notas del salón.*)*

Capítulo 3: Solución Final

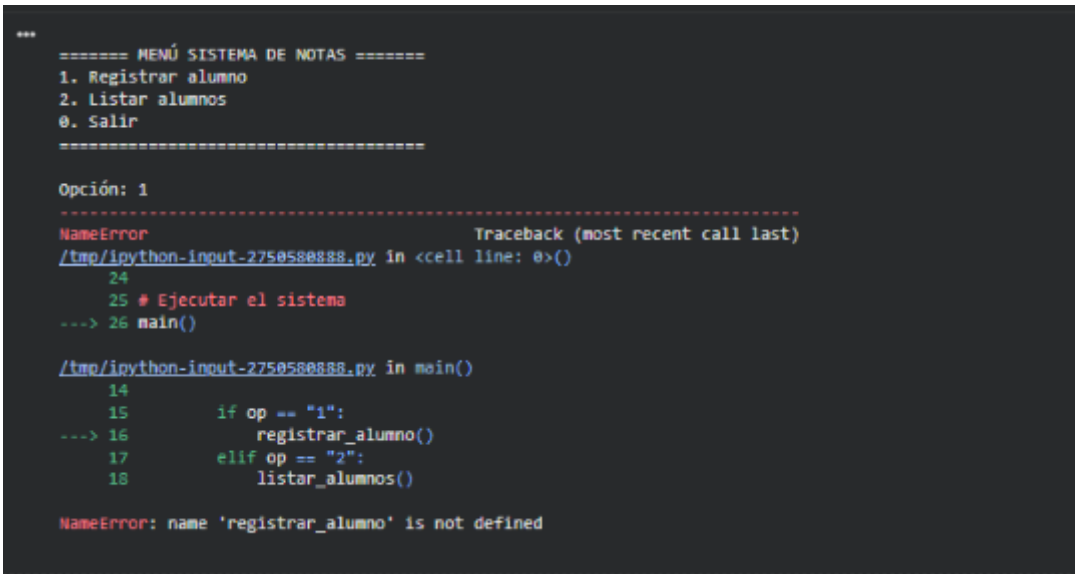
1. **Código limpio, bien comentado y estructurado.**

```python
import json

# ================= SISTEMA DE NOTAS =================

class Alumno:
    def __init__(self, aid, nombre, nota1, nota2, nota3):
        self.aid = aid
        self.nombre = nombre
        self.nota1 = nota1
        self.nota2 = nota2
        self.nota3 = nota3
        self.promedio = round((nota1 + nota2 + nota3) / 3, 2)

    def a_dict(self):
        return {
            "aid": self.aid,
            "nombre": self.nombre,
            "nota1": self.nota1,
            "nota2": self.nota2,
            "nota3": self.nota3,
            "promedio": self.promedio
        }

    @staticmethod
    def desde_dict(d):
        return Alumno(d["aid"], d["nombre"], d["nota1"], d["nota2"], d["nota3"])


# Lista de alumnos
alumnos = []
siguiente_id = 1
archivo = "alumnos.json"


# ================= OPCIÓN 1: REGISTRAR =================

def registrar_alumno():
```

```
    global siguiente_id

    print("\n--- Registrar alumno ---")
    nombre = input("Nombre del alumno: ")

    try:
        n1 = float(input("Nota 1: "))
        n2 = float(input("Nota 2: "))
        n3 = float(input("Nota 3: "))
    except ValueError:
        print(" Error: Debes ingresar números.")
        return

    nuevo = Alumno(siguiente_id, nombre, n1, n2, n3)
    alumnos.append(nuevo)

    print(f"✓ Alumno '{nuevo.nombre}' registrado con ID {nuevo.aid}. Promedio: {nuevo.promedio}")

    siguiente_id += 1


# ================= OPCIÓN 2: LISTAR =================

def listar_alumnos():
    print("\n--- Lista de alumnos ---")
    if len(alumnos) == 0:
        print("No hay alumnos registrados.")
        return

    for a in alumnos:
        print(f"ID {a.aid} | {a.nombre} | Notas: {a.nota1}, {a.nota2}, {a.nota3} | Promedio: {a.promedio}")


# ================= OPCIÓN 3: ELIMINAR =================

def eliminar_alumno():
    if len(alumnos) == 0:
        print("No hay alumnos para eliminar.")
        return

    try:
        aid = int(input("Ingrese el ID del alumno a eliminar: "))
    except:
        print(" Debes ingresar un número.")
        return

    for a in alumnos:
        if a.aid == aid:
            alumnos.remove(a)
            print(f"🗑 Alumno con ID {aid} eliminado.")
            return

    print(" No existe un alumno con ese ID.")


# ================= OPCIÓN 4: GUARDAR =================

def guardar_datos():
    data = [a.a_dict() for a in alumnos]

    with open(archivo, "w") as f:
        json.dump({
            "siguiente_id": siguiente_id,
            "alumnos": data
        }, f, indent=4)

    print(" Datos guardados correctamente.")


# ================= MENÚ PRINCIPAL =================

def mostrar_menu():
    print("""
======= MENÚ SISTEMA DE NOTAS =======
1. Registrar alumno
2. Listar alumnos
3. Eliminar alumno
```

```
4. Guardar datos
0. Salir
==================================
""")

def main():
    while True:
        mostrar_menu()
        op = input("Opción: ")

        if op == "1":
            registrar_alumno()

        elif op == "2":
            listar_alumnos()

        elif op == "3":
            eliminar_alumno()

        elif op == "4":
            guardar_datos()

        elif op == "0":
            print(" Saliendo del sistema...")
            break

        else:
            print(" Opción inválida.")


# Ejecutar
main()
```

2. **Capturas de pantalla de las ventanas de ejecución con las diversas pruebas de validación de datos**



```
...
    ======= MENÚ SISTEMA DE NOTAS =======
    1. Registrar alumno
    2. Listar alumnos
    0. Salir
    ==================================

    Opción: 1
    ---------------------------------------------------------------------------
    NameError                                 Traceback (most recent call last)
    /tmp/ipython-input-2750580888.py in <cell line: 0>()
         24
         25 # Ejecutar el sistema
    ---> 26 main()

    /tmp/ipython-input-2750580888.py in main()
         14
         15         if op == "1":
    ---> 16             registrar_alumno()
         17         elif op == "2":
         18             listar_alumnos()

    NameError: name 'registrar_alumno' is not defined
```

```
======= MENÚ SISTEMA DE NOTAS =======
1. Registrar alumno
2. Listar alumnos
0. Salir
====================================

Opción: 1

--- Registrar alumno ---
Nombre del alumno: Jhojand
Nota 1: 12
Nota 2: 13
Nota 3: 14
✓ Alumno 'Jhojand' registrado con ID 1. Promedio: 13.0

======= MENÚ SISTEMA DE NOTAS =======
1. Registrar alumno
2. Listar alumnos
0. Salir
====================================

Opción: 2

--- Lista de alumnos ---
ID 1 | Jhojand | Notas: 12.0, 13.0, 14.0 | Promedio: 13.0

======= MENÚ SISTEMA DE NOTAS =======
1. Registrar alumno
2. Listar alumnos
0. Salir
====================================

Opción: 0
 Saliendo del sistema...
```

[3] 0s

```
119                 eliminar_alumno()
120
121         elif op == "0":
122             print(" Saliendo del sistema...")
123             break
124
125         else:
126             print(" Opción inválida.")
127
128
129 # Ejecutar
130 main()
131
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/tmp/ipython-input-3933789706.py in <cell line: 0>()
    128
    129 # Ejecutar
--> 130 main()

                                 2 frames
/usr/local/lib/python3.12/dist-packages/ipykernel/kernelbase.py in _input_request(self, prompt, ident, parent, password)
   1217             except KeyboardInterrupt:
   1218                 # re-raise KeyboardInterrupt, to truncate traceback
-> 1219                 raise KeyboardInterrupt("Interrupted by user") from None
   1220             except Exception:
   1221                 self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```

```
145 main()
146
```

```
  File "/tmp/ipython-input-3317888714.py", line 100
    "siguiente_
    ^
SyntaxError: unterminated string literal (detected at line 100)
```

Próximos pasos: Explicar error

3. Manual de usuario – sistema de notas

1. Iniciar el sistema

Al ejecutar el programa aparece un menú con todas las opciones disponibles.

2. Registrar alumno (Opción 1)

- Escribe el nombre del alumno.
- Ingresa las tres notas.
- El sistema calcula el promedio y asigna un ID automáticamente.

3. Listar alumnos (Opción 2)

- Muestra todos los alumnos registrados.
- Incluye ID, nombre, notas y promedio.

4. Eliminar alumno (Opción 3)

- Ingresa el ID del alumno que deseas borrar.
- Si existe, será eliminado de la lista.

5. Guardar datos (Opción 4)

- Guarda todos los alumnos en un archivo JSON.
- Permite mantener la información para usarla después.

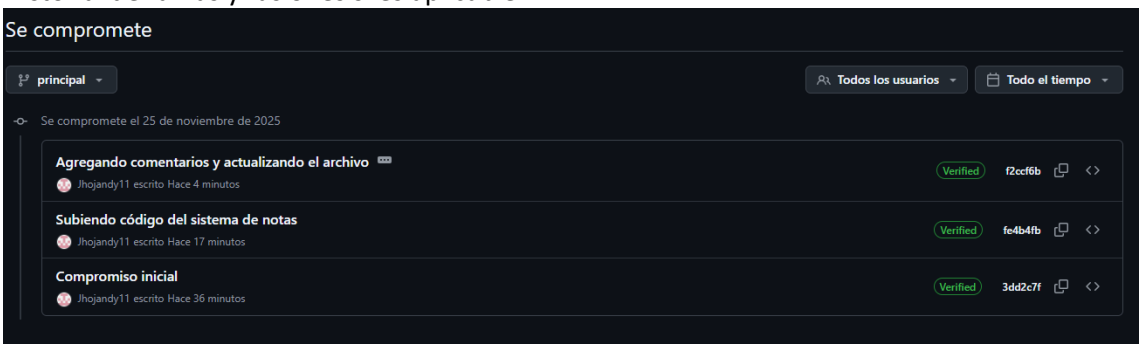
6. Salir (Opción 0)

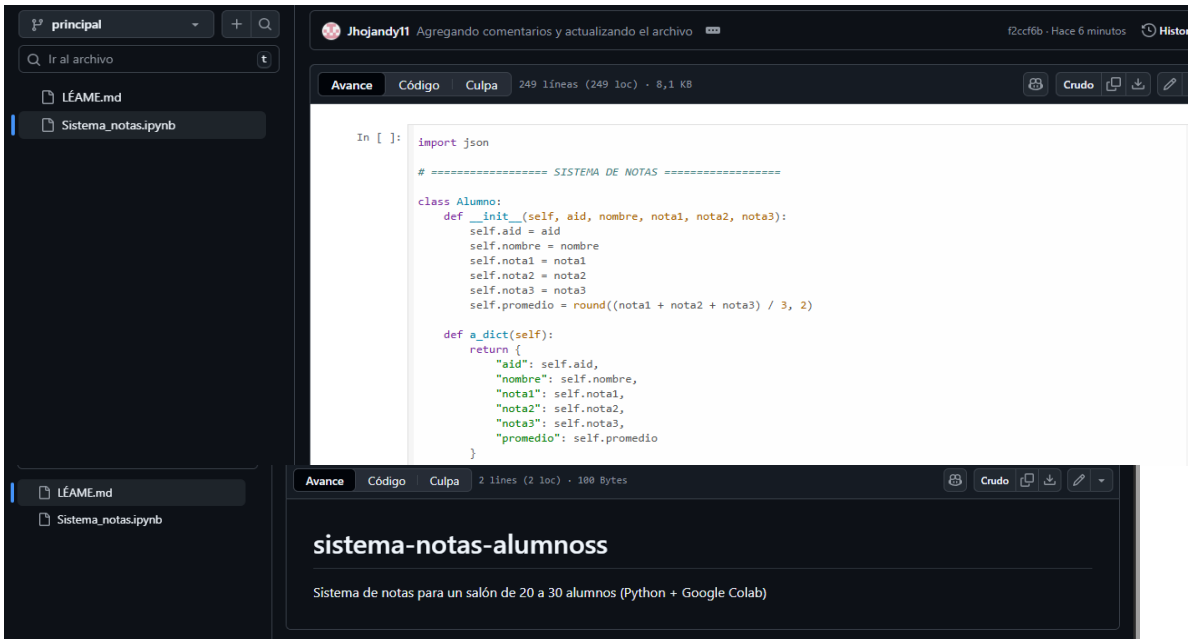
- Finaliza la ejecución del sistema.

Capítulo 4: Evidencias de Trabajo en Equipo

1. **Repositorio con Control de Versiones (Capturas de Pantalla)**
   - Registro de commits claros y significativos que evidencien aportes individuales (proactividad).

   - Historial de ramas y fusiones si es aplicable.





- Enlace a la herramienta colaborativa

https://github.com/Jhojandy11/sistema-notas-alumnoss