MAESTRÍA EN CIENCIA DE DATOS

UNIVERSIDAD NACIONAL ARTURO JAURECHE

TRABAJO INTEGRADOR

MATERIA: PROGRAMACIÓN (MI5502)

AUTORES

Victor Conti

Deborah Herrera

Franco Manini

Julia Fernandez

PROFESORES

Jorge Rafael Osio

Nahuel Christian Botta

AÑO

2024

Índice general

Índice de figuras

1. Introducción

Este trabajo práctico grupal tiene como objetivo aplicar los conceptos aprendidos en el curso de Programación a un proyecto concreto.

En el ámbito educativo, el seguimiento y análisis del rendimiento académico de los estudiantes es una tarea fundamental para evaluar su progreso y tomar decisiones respecto al desempeño individual, por ejemplo. En este sentido, el presente proyecto tiene como objetivo desarrollar un sistema de gestión académica que facilite el procesamiento de las calificaciones obtenidas por los alumnos en tres asignaturas diferentes, aunque puede ser ampliado fácilmente a más asignaturas.

Este sistema informático está diseñado para realizar diversos cálculos y operaciones relacionadas con las notas de los estudiantes. En primer lugar, permite obtener el promedio de calificaciones de cada alumno, lo cual brinda una visión general de su desempeño académico. Además, determina de manera automática la situación académica de cada individuo, es decir, si ha aprobado o reprobado las asignaturas en cuestión.

Más allá de los cálculos individuales, el sistema también ofrece información estadística a nivel global. Permite calcular el promedio general de todas las notas ingresadas, así como el promedio específico para cada una de las tres asignaturas. Estos datos agregados serán de gran utilidad para los docentes y autoridades educativas, ya que les permitirán evaluar el nivel general de la clase y detectar posibles áreas de mejora.

Asimismo, el sistema cuenta con funcionalidades para la visualización y gestión de la información de los estudiantes, facilitando el acceso a sus datos personales y académicos de manera organizada y eficiente. El algoritmo, desarrollado en el lenguaje de programación *Python*, está disponible en la plataforma Github [1] permitiendo que otras personas contribuyan o lo modifiquen según sus necesidades.

El proyecto busca brindar una herramienta integral que optimice los procesos de gestión académica, permitiendo un análisis detallado del rendimiento estudiantil y proporcionando información valiosa para la toma de decisiones en el ámbito educativo.

[1] https://github.com/VictorContiDesign/mi5502_TFI

2. Desarrollo

Se describe aquí el diagrama de flujo y la lógica que se ha utilizado para desarrollar la aplicación.

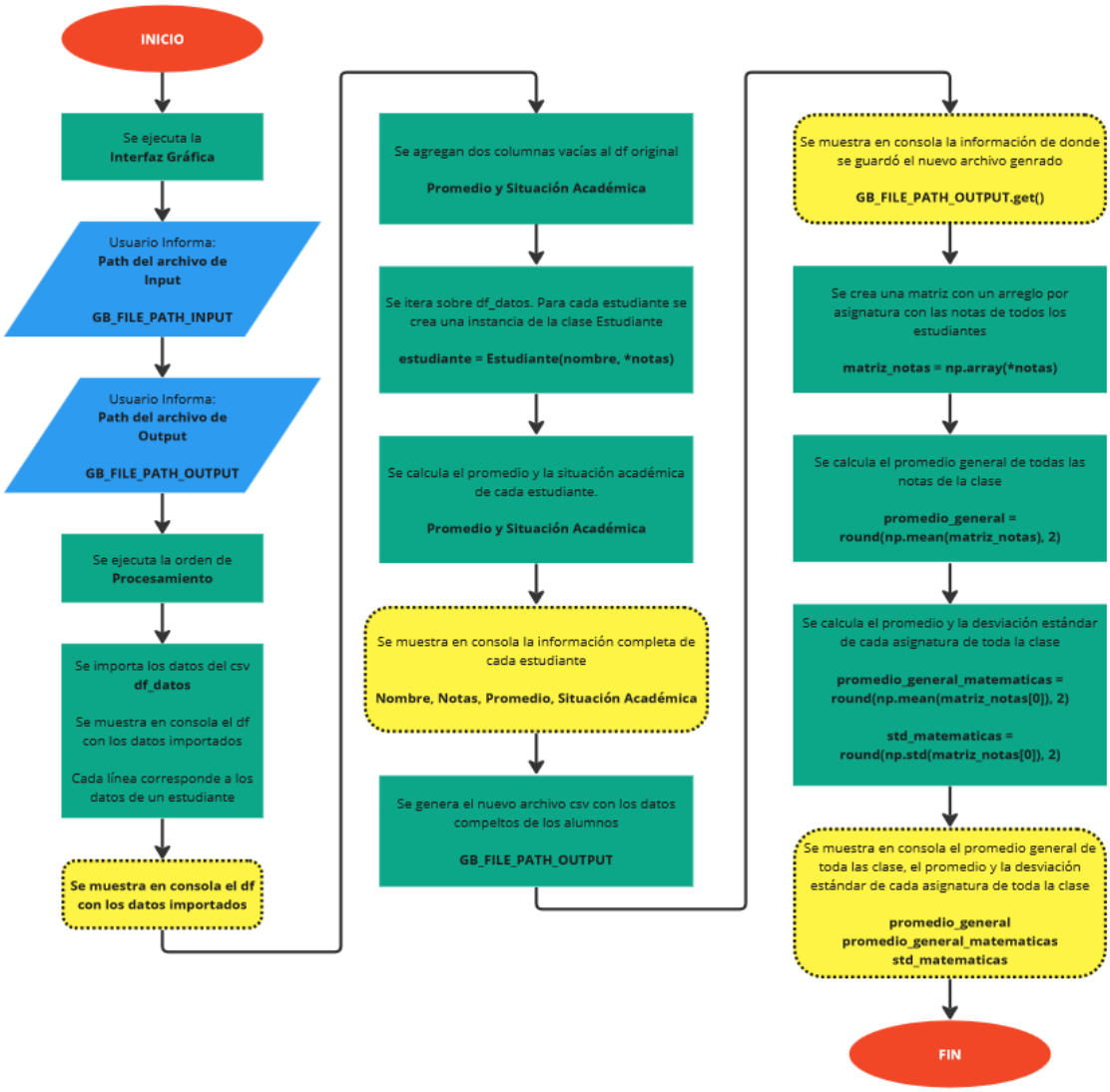

Figura 1: Diagrama de flujo de la aplicación.

3. Resultados

3.1. Introducción

En esta sección del informe, presentamos los resultados obtenidos de la ejecución de nuestro código diseñado para gestionar y analizar las calificaciones de los estudiantes en las asignaturas de matemáticas, ciencias y lenguaje.

Continuando con la exposición de los resultados, es relevante destacar como la interfaz gráfica de usuario se convierte en un componente crucial de nuestra solución, proporcionando una herramienta accesible para la selección y generación de archivos de salida (Fig. 4). Además, con los resultados obtenidos a través del programa, hemos podido confirmar la efectividad de nuestro código en el procesamiento y análisis de calificaciones. La salida generada demuestra que nuestro programa no sólo realiza cálculos precisos del promedio de calificaciones de los estudiantes en asignaturas especificas, sino que también clasifica con éxito su situación académica.

El archivo CSV resultante ofrece una visión clara del rendimiento académico como se observa en la Fig. 24. Además de las calificaciones númericas, el programa ha asignado correctamente el estado de 'Aprobado/a' a cada estudiante, en caso que la nota sea mayor a 6 puntos, demostrando que la lógica de evaluación funciona correctamente en el programa.

3.2. Proceso

A continuación se describe el proceso completo de creación del código y de los resultados obtenidos, tanto en consola como en el archivo CSV resultante.

El punto de entrada del programa consiste en crear una ventana gráfica, configurar y posicionar todos sus elementos, es decir, los labels, las entradas de texto, los botones y sus acciones derivadas, así como el logo y el título de la ventana, tal como se ve en la Fig. 2. En las líneas 26, 34 y 40 se observa que a cada botón se le asigna en el parámetro 'command' la ejecución de una función que está desarrollada en otra parte del código. Estas funciones se explicarán más adelante.

```
1  ########
2  # MAIN #
3  ########
4
5  if __name__ == "__main__":
6
7      # Crear la ventana de la interfaz gráfica
8      root = ctk.CTk()
9      root.title("Calificaciones Estudiantes")
10
11     # Centrar la ventana en pantalla
12     screen_width = root.winfo_screenwidth()
13     screen_height = root.winfo_screenheight()
14     x_pos = screen_width // 2 - 180
15     y_pos = screen_height // 2 - 180
16     root.geometry(f"+{x_pos}+{y_pos}")
17
18     # Crear un marco para agrupar los elementos
19     frame = ctk.CTkFrame(root)
20     frame.pack(pady=20, padx=20)
21     frame.update()
22
23     # Seleccionar Archivo de Entrada
24     label_file_input = ctk.CTkLabel(frame, text="Seleccionar Archivo a Leer:")
25     label_file_input.pack()
26     button_file_input = ctk.CTkButton(frame, text="Examinar...", command = buscar_archivo_entrada)
27     button_file_input.pack(pady=10)
28     GB_FILE_PATH_INPUT = ctk.CTkEntry(frame, width=300)
29     GB_FILE_PATH_INPUT.pack(padx=10, pady=20)
30
31     # Definir Archivo de Salida
32     label_file_output = ctk.CTkLabel(frame, text="Definir Archivo a Generar:")
33     label_file_output.pack()
34     button_file_output = ctk.CTkButton(frame, text="Guardar cómo...", command = definir_archivo_salida)
35     button_file_output.pack(pady=10)
36     GB_FILE_PATH_OUTPUT = ctk.CTkEntry(frame, width=300)
37     GB_FILE_PATH_OUTPUT.pack(padx=10)
38
39     # Procesamiento del archivo de entrada
40     button_process = ctk.CTkButton(frame, text="EJECUTAR \nPROCESO", command = ejecutar_proceso)
41     button_process.pack(pady=40)
42
43     # Iniciar bucle principal
44     root.mainloop()
```

Figura 2: Punto de entrada del programa. Construcción de la ventana gráfica.

En la Fig. 4 se observa la ventana gráfica que se construye cuando se ejecuta en consola el archivo *main.py* (Fig. 3)

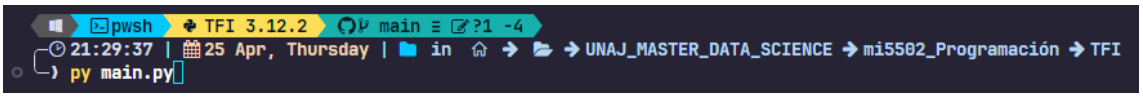

Figura 3: Ejecución del archivo main

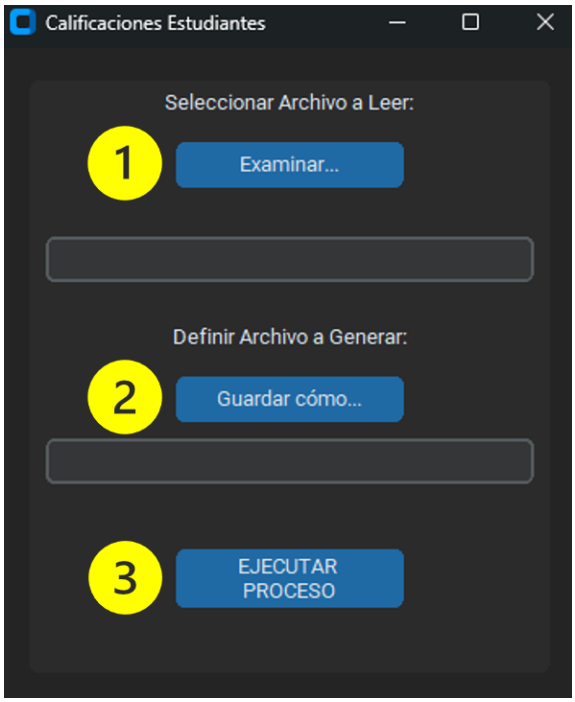

Figura 4: Interfaz de usuario que muestra las opciones para cargar el archivo de entrada, establecer el archivo de salida y ejecutar el procesamiento de datos.

El primer botón llama a una función (Fig. 5) que abre una ventana de diálogo (Fig. 6), la cual permite buscar el archivo de input donde está la lista de estudiantes con sus notas (Fig. 7).

```
1  def buscar_archivo_entrada():
2      global GB_FILE_PATH_INPUT
3      file_path_input = filedialog.askopenfilename()
4      GB_FILE_PATH_INPUT.delete(0, ctk.END)
5      GB_FILE_PATH_INPUT.insert(0, file_path_input)
6      print("**********")
7      print("Archivo de Entrada: ", GB_FILE_PATH_INPUT.get())
8      print("**********")
```

Figura 5: Función que busca el archivo de input.

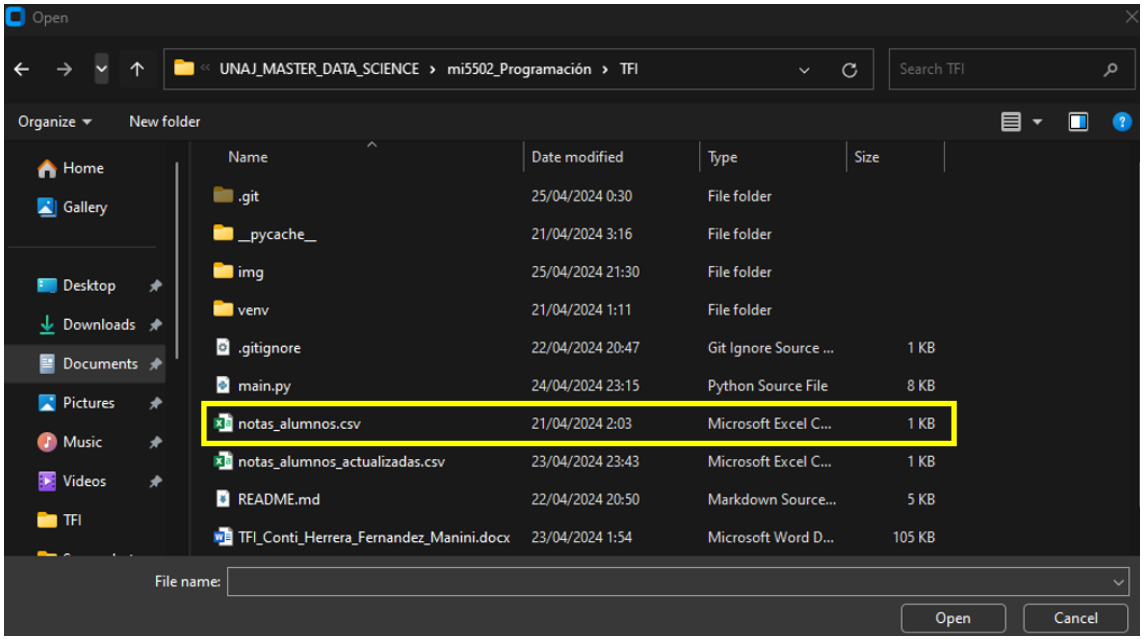

Figura 6: Cuadro de diálogo que busca el archivo de input

Estudiante	Matematicas	Ciencias	Lenguaje
Victor Conti	7.0	7.0	9.0
Deborah Herrera	8.5	9.7	7.0
Julia Fernandez	9.0	9.8	7.0
Franco Manini	9.0	9.4	7.0

Figura 7: Archivo de Input

El segundo botón llama a una función (Fig. 8) que abre una ventana de diálogo (Fig. 9) que permite decidir dónde se va a guardar el archivo que se va a generar al final del proceso.

```
def definir_archivo_salida():
    global GB_FILE_PATH_OUTPUT
    file_path_output = filedialog.asksaveasfilename(initialfile = "notas_alumnos_actualizadas.csv", title = "Select file", filetypes = (("csv files","*.csv"),))
    GB_FILE_PATH_OUTPUT.delete(0, ctk.END)
    GB_FILE_PATH_OUTPUT.insert(0, file_path_output)
    print("**********")
    print("Archivo de Salida: ", GB_FILE_PATH_OUTPUT.get())
    print("**********")
```

Figura 8: Función que guarda el archivo de Output

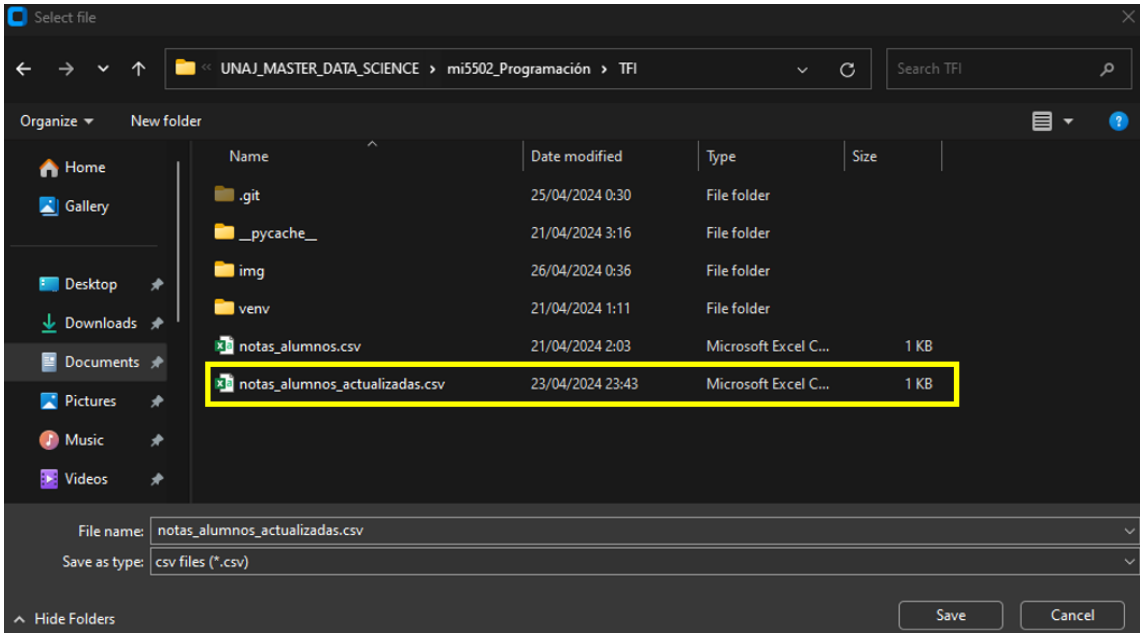

Figura 9: Cuadro de diálogo para guardar archivo de Output

El tercer botón permite ejecutar el proceso completo. La función asociada, que se observa en la figura 10, importa los datos del archivo CSV de entrada utilizando la clase **ImportarDesdeArchivo** (Fig. 11), los guarda en forma de dataframe, y los imprime en consola (Fig. 13).

```
def ejecutar_proceso():
    global GB_FILE_PATH_INPUT
    global GB_FILE_PATH_OUTPUT

    # Importamos los datos generales de los estudiantes desde el archivo csv
    datos = ImportarDesdeArchivo(GB_FILE_PATH_INPUT)
    df_datos = datos.getDatos()

    # Mostramos en consola los datos importados
    print()
    print(colored("DATOS GENERALES IMPORTADOS DE LOS ESTUDIANTES:", "yellow"))
    print(df_datos)
    print()
```

Figura 10: Se importan los datos y se muestran en consola

```python
class ImportarDesdeArchivo:
    def __init__(self, path_archivo):
        self.path_archivo = path_archivo

    def getDatos(self):
        datos = pd.read_csv("notas_alumnos.csv", sep=";")
        return datos
```

Figura 11: Clase para importar archivos CSV

```
DATOS GENERALES IMPORTADOS DE LOS ESTUDIANTES:
        Estudiante  Matematicas  Ciencias  Lenguaje
0     Victor Conti          7.0       7.0       9.0
1  Deborah Herrera          8.5       9.7       7.0
2  Julia Fernandez          9.0       9.8       7.0
3    Franco Manini          9.0       9.4       7.0
```

Figura 12: Datos Importados

A las cuatro columnas de este dataframe se le agregan dos nuevas columnas con contenido vacío (Promedio y Situación Académica), que serán completadas luego de las operaciones y cálculos necesarios (Fig. 13).

```python
# Agregamos al dataframe las columnas "Promedio" y "Situacion_Academica"
# Para generar después un nuevo archivo csv con la información completada
df_datos["Promedio"] = ""
df_datos["Situacion_Academica"] = ""
```

Figura 13: Inclusión de columnas Promedio y Situación Académica

El proceso itera sobre cada línea del dataframe con los datos de los estudiantes, extrayendo sus nombres y sus notas. Con esta información se calcula el promedio y se define la situación académica para cada estudiante. Además se crea una instancia de la clase 'Estudiante' para cada estudiante (Fig. 14).

Se agrega al dataframe los promedios y situaciones académicas de cada estudiante (Fig. 14), y se imprime en consola la información recopilada y calculada para cada uno (Fig. 15), creando para esto la clase **Estudiante** (Fig. 16).

```python
# Generamos una instancia de la clase Estudiante para cada estudiante
for index, row in df_datos.iterrows():

    # Aislamos los datos de cada Estudiante para generar la instancia
    nombre = df_datos["Estudiante"][index]
    nota_matematica = df_datos["Matematicas"][index]
    nota_ciencas = df_datos["Ciencias"][index]
    nota_lenguaje = df_datos["Lenguaje"][index]
    estudiante = Estudiante(nombre, nota_matematica, nota_ciencas, nota_lenguaje)

    # Calculamos el promedio y definimos la situación académica de cada estudiante
    promedio = estudiante.calcular_promedio
    situacion_academica = estudiante.situacion_academica

    # Mostramos la información en consola
    print(colored(f"ESTUDIANTE n°{index + 1}", "yellow"))
    print(f"    > {estudiante}")
    print(f"    > Promedio: {promedio:.2f}")
    print(f"    > Situación Académica: {situacion_academica}")
    print()

    # Actualizamos los datos generales de los estudiantes
    # Agregamos el promedio y la situación académica
    df_datos.loc[index, "Promedio"] = round(promedio, 2)
    df_datos.loc[index, "Situacion_Academica"] = situacion_academica
```

Figura 14: Creación de una instancia Estudiante para cada estudiante

```
ESTUDIANTE n°1
    > Nombre: Victor Conti - Matematicas: 7.0 - Ciencias: 7.0 - Lenguaje: 9.0
    > Promedio: 7.67
    > Situación Académica: Aprobado/a

ESTUDIANTE n°2
    > Nombre: Deborah Herrera - Matematicas: 8.5 - Ciencias: 9.7 - Lenguaje: 7.0
    > Promedio: 8.40
    > Situación Académica: Aprobado/a

ESTUDIANTE n°3
    > Nombre: Julia Fernandez - Matematicas: 9.0 - Ciencias: 9.8 - Lenguaje: 7.0
    > Promedio: 8.60
    > Situación Académica: Aprobado/a

ESTUDIANTE n°4
    > Nombre: Franco Manini - Matematicas: 9.0 - Ciencias: 9.4 - Lenguaje: 7.0
    > Promedio: 8.47
    > Situación Académica: Aprobado/a
```

Figura 15: Información en consola de cada estudiante

Se imprime en consola el dataframe completado con los promedios y las situaciones académicas (Fig. 17 y Fig. 18).

Se guarda el dataframe completo en el archivo de formato CSV cuya ruta se ha especificado antes (Fig. 19).

```python
class Estudiante:
    def __init__(self, nombre, nota_matematica, nota_ciencias, nota_lenguaje):
        self.nombre = nombre
        self.nota_matematica = nota_matematica
        self.nota_ciencias = nota_ciencias
        self.nota_lenguaje = nota_lenguaje

    def __str__(self):
        return f"Nombre: {self.nombre} - Matematicas: {self.nota_matematica} - Ciencias: {self.nota_ciencias} - Lenguaje: {self.nota_lenguaje}"

    @property
    def calcular_promedio(self):
        self.promedio = (self.nota_matematica + self.nota_ciencias + self.nota_lenguaje) / 3
        return self.promedio

    @property
    def situacion_academica(self):
        if self.promedio >= 6:
            return "Aprobado/a"
        else:
            return "Reprobado/a"
```

Figura 16: Clase Estudiante

```python
# Mostramos en consola los datos modificados
print(colored("DATOS GENERALES ACTUALIZADOS DE LOS ESTUDIANTES:", "yellow"))
print(df_datos)
print()
```

Figura 17: Impresión en consola el dataframe completo

```
DATOS GENERALES ACTUALIZADOS DE LOS ESTUDIANTES:
        Estudiante  Matematicas  Ciencias  Lenguaje Promedio Situacion_Academica
0     Victor Conti          7.0       7.0       9.0     7.67          Aprobado/a
1  Deborah Herrera          8.5       9.7       7.0      8.4          Aprobado/a
2  Julia Fernandez          9.0       9.8       7.0      8.6          Aprobado/a
3    Franco Manini          9.0       9.4       7.0     8.47          Aprobado/a
```

Figura 18: Visualización en consola el dataframe completo

```python
# Guardamos en un archivo csv los datos modificados
df_datos.to_csv(GB_FILE_PATH_OUTPUT.get(), index=False, sep=";")

# Mensaje de confirmación de final de proceso en consola
print(f"El archivo de resumen de la evaluación de los estudiantes se ha guardado en: ")
print(colored(f"    > {GB_FILE_PATH_OUTPUT.get()}", "light_green"))
print()
```

Figura 19: Guardado del dataframe en un archivo CSV

Se visualiza en consola la ruta donde se ha guardado el archivo (Fig. 20).

```
El archivo de resumen de la evaluación de los estudiantes se ha guardado en:
    > C:/Users/victo/OneDrive/Documents/_ESTUDIOS/0_UNAJ_DATA_SCIENCE/UNAJ_MASTER_DATA_SCIENCE/mi5502_Programación/TFI/notas_alumnos_actualizadas.csv
```

Figura 20: Ruta del archivo CSV guardado

Se crea una matriz en la que cada una de sus líneas son un arreglo compuesto por las notas de todos los alumnos para cada asignatura. De esta manera se puede calcular el promedio general de todas las notas de la clase (Fig. 21).

Se calcula el promedio y la desviación estándar de toda la clase para cada asignatura (Fig.

```python
# Creamos una matriz con un arreglo por asignatura con las notas de todos los estudiantes
matriz_notas = np.array([
    df_datos["Matematicas"],
    df_datos["Ciencias"],
    df_datos["Lenguaje"],
])

# Calculamos el promedio general de todas las notas de la clase
promedio_general = round(np.mean(matriz_notas), 2)
print(colored(f"El promedio general de todas las notas es de: {promedio_general}", "light_green"))
print()
```

Figura 21: Cálculo del promedio general de la clase mediante el uso de una matriz

22).

```python
# Calculamos el promedio y la desviación estándar de cada asignatura de toda la clase
promedio_general_matematicas = round(np.mean(matriz_notas[0]), 2)
std_matematicas = round(np.std(matriz_notas[0]), 2)
print(colored(f"El promedio general de Matemáticas es de: {promedio_general_matematicas}. Su desviación estándar es: {std_matematicas}.", "light_green"))
print()
promedio_general_ciencias = round(np.mean(matriz_notas[1]), 2)
std_ciencias = round(np.std(matriz_notas[1]), 2)
print(colored(f"El promedio general de Ciencias es de: {promedio_general_ciencias}. Su desviación estándar es: {std_ciencias}.", "light_green"))
print()
promedio_general_lenguaje = round(np.mean(matriz_notas[2]), 2)
std_lenguaje = round(np.std(matriz_notas[2]), 2)
print(colored(f"El promedio general de Lenguaje es de: {promedio_general_lenguaje}. Su desviación estándar es: {std_lenguaje}.", "light_green"))
print()
```

Figura 22: Cálculo del promedio y desviación estándar para cada materia

Se puede observar en consola el resultado del promedio general y de los promedios de toda la clase para cada asignatura (Fig. 23).

```
El promedio general de todas las notas es de: 8.28

El promedio general de Matemáticas es de: 8.38. Su desviación estándar es: 0.82.

El promedio general de Ciencias es de: 8.98. Su desviación estándar es: 1.15.

El promedio general de Lenguaje es de: 7.5. Su desviación estándar es: 0.87.
```

Figura 23: Resultados de promedios y desviaciones estándar visualizados en consola

Finalmente se abre el archivo CSV generado donde se muestran las calificaciones finales de los estudiantes en matemáticas, ciencias y lenguaje, junto con sus promedios y la situación académica resultante (Fig. 24).

Estudiante	Matematicas	Ciencias	Lenguaje	Promedio	Situacion_Academica
Victor Conti	7.0	7.0	9.0	7.67	Aprobado/a
Deborah Herrera	8.5	9.7	7.0	8.4	Aprobado/a
Julia Fernandez	9.0	9.8	7.0	8.6	Aprobado/a
Franco Manini	9.0	9.4	7.0	8.47	Aprobado/a

Figura 24: Archivo CSV generado

4. Conclusiones

El presente trabajo práctico tuvo como objetivo desarrollar una aplicación de escritorio capaz de procesar las calificaciones de los estudiantes de un curso, a partir de un archivo CSV con las notas de cada alumno en tres asignaturas 'matemáticas, ciencias y lenguaje'. Mediante la integración de diversas librerías y conceptos de programación en Python, se logró crear una herramienta funcional que aporta valor en el ámbito educativo.

La aplicación permite calcular de manera automatizada los promedios individuales de cada estudiante, determinar su situación académica (aprobado o reprobado) según un criterio preestablecido, y generar un nuevo archivo CSV con esta información adicional. Además, realiza un análisis estadístico básico de las calificaciones, calculando los promedios generales y las desviaciones estándar por asignatura para toda la clase.

La implementación de una interfaz gráfica de usuario intuitiva y amigable, utilizando la librería *customtkinter*, facilita la interacción con la aplicación, permitiendo al usuario seleccionar el archivo CSV de entrada y definir la ubicación y nombre del archivo de salida de manera sencilla.

Este trabajo práctico integrador demuestra la capacidad de aprovechar diversas herramientas y conceptos de programación en Python para desarrollar una solución práctica y escalable en el ámbito educativo. La aplicación agiliza el procesamiento de calificaciones, brinda información valiosa sobre el desempeño de los estudiantes y facilita el análisis estadístico de los resultados académicos.

Si bien la aplicación cumple con los requisitos planteados, existen oportunidades para su mejora y expansión futura. Algunas posibles mejoras incluyen la incorporación de funcionalidades adicionales, como la generación de informes más detallados, la visualización gráfica de los resultados y la integración con sistemas de gestión de calificaciones existentes.