ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

ALGORITMOS Y ESTRUCTURAS DE DATOS

TDSD

ASIGNATURA:	ALGORITMOS Y ESTRUCTURAS DE DATOS
PROFESOR:	Ing. Lorena Chulde
PERÍODO ACADÉMICO:	2023-B

PRUEBA – BIMESTRE 2

Nombre del estudiante:

Guerra Lovato Josué Eduard

Soria Ansa Richard Mauricio

Pérez Orosco Carlos David

2023-B

Contenido

1. GUÍA DE PROYECTO

Aprendiendo del Mundo Real

Se requiere desarrollar un Sistema de Gestión de Calificaciones, para un Instituto de Educación Superior.

Una vez que se han finalizado las respectivas reuniones con el Product Owner, se han determinado los siguientes requerimientos:

- Desarrollar el módulo del docente.
- El usuario docente debe iniciar sesión con las siguientes credenciales:
  - Usuario: docente@esfot.edu.ec
  - Contraseña: Docente2023*
- Almacenar la información de los archivos en una Base de datos llamada "**reportes**"
- Un menú repetitivo de opciones que realice lo siguiente:

1) Si el profesor ingresa a la opción 1, el sistema debe permitirle:
   a. Registrar los datos del profesor, materia, estudiantes y sus calificaciones respectivas.
   b. Guardar la información en un archivo plantilla "calificaciones.txt".

2) Si el profesor ingresa a la opción 2, el sistema debe permitirle:
   a. Ordenar las calificaciones de los estudiantes en base a un algoritmo de ordenamiento que el docente debe seleccionar (Burbuja, Inserción, Selección, MergeSort, QuickSort, etc.).
   b. Almacenar los resultados en un archivo plantilla "ordenamiento.txt"
   c. Mostrar la información del archivo en la consola del programa.

3) Si el profesor ingresa a la opción 3, el sistema debe permitirle:
- **a.** Buscar una calificación, el cual debe ingresar la calificación y luego seleccionar con qué tipo de algoritmo de búsqueda (Lineal, Binaria, Interpolación etc.) lo desea realizar la búsqueda.
- **b.** Almacenar los resultados en un archivo plantilla "búsqueda.txt"
- **c.** Mostrar la información del archivo en la consola del programa.

4) Si el profesor ingresa a la opción 3, el sistema debe permitirle:
- **a.** Salir del sistema

- El sistema debe abarcar todos los temas tratados en clase.
- El sistema puede utilizar un GUI llamado TKinter. (Opcional)

Archivo "calificaciones.txt"

COLEGIO o UNIVERSIDAD _____
REPORTE DE CALIFICACIONES

Año lectivo o Semestre: _____

Materia: _____

N°-	Estudiante	Apellido	correo	Nota 1	Nota 2	Total
1	Juan	Alvear				18
2	Pedro	Sarmiento				15
3	Cesar	Cajas				14
4	María	Soto				12
…..	….					….

RESUMEN		
Promedio del curso:	14.75	
Total	**Aprobados** (14 - 20)	3
	Suspenso (09 - 13)	1
	Reprobados (01 - 08)	0

_____
Docente
_____

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Archivo "ordenamiento.txt"

COLEGIO o UNIVERSIDAD _____

REPORTE DE CALIFICACIONES

Calificaciones Ordenadas

ALGORITMO: Burbuja / Heap Sort / Merge Sort / Quick Sort / ……

N°- |10|-|14|-|17|-|19|-…… (Mostrar las calificaciones ordenadas)

_____

Docente

_____

_____

Archivo "buscar.txt"

COLEGIO o UNIVERSIDAD _____

REPORTE DE CALIFICACIONES

Búsqueda de Calificaciones

ALGORITMO: Lineal / Binaria / Interpolación / ……

La calificación a buscar fue de: 15

Corresponde al estudiante:

Byron Loarte byron@hotmail.com ……

_____

Docente

_____

_____

Felicitaciones querid@s estudiantes han logrado trabajar en equipo y aplicar a un ejemplo del mundo real.

Finalmente, se debe presentar la documentación con la información generada en los puntos anteriores, además de un índice de tablas, figuras. El cual debe ser subido a la plataforma virtual con el nombre del archivo **AED-GRUPO#-PROYECTOFINAL**.

2. EXPOSICIÓN DEL PROYECTO

El grupo conformado realizara una presentación de su proyecto abarcando los siguientes puntos:

1. Los miembros del grupo deberán exponer toda las (arquitectura, herramientas, archivos de configuración, paquetes instalados, etc..), utilizadas para la configuración y puesta en marcha del sistema.
2. La funcionalidad debe presentarse en un vídeo (5-10min) el cual debe estar alojado en YouTube.
3. Ser lo más concreto posible ya que el tiempo, la calidad de la presentación y exposición determinaran la nota final.

3. INDICACIONES GENERALES

- El día 01 de marzo, siendo la última clase del semestre se reserva para entrega de la documentación final y la presentación del sistema.
- El trabajo será evaluado de acuerdo a la rúbrica propuesta.
- Finalmente cabe recordar que la nota del proyecto final es grupal por ende el día de la presentación se procederá a una ronda de preguntas donde los integrantes del grupo demostraran los conocimientos adquiridos a lo largo del semestre y de la materia.

4. PRESENTACIÓN DEL PROYECTO

a. *Herramientas a usar:*

Python:

Python es un lenguaje de programación de alto nivel, interpretado, de propósito general y multiplataforma. Fue creado a finales de los años 80 y principios de los 90 por Guido van Rossum y ha crecido hasta convertirse en uno de los lenguajes de programación más populares del mundo.

Algunas características clave de Python incluyen su sintaxis clara y legible, que favorece la legibilidad del código y la productividad del programador. Es un lenguaje multiparadigma, lo que significa que soporta diferentes estilos de programación, como la programación orientada a objetos, la programación imperativa y la programación funcional.

Python tiene una amplia gama de aplicaciones, desde desarrollo web y scripting hasta inteligencia artificial, análisis de datos, desarrollo de juegos y más. Su comunidad activa y su amplia disponibilidad de bibliotecas y frameworks hacen que sea una opción popular tanto para principiantes como para desarrolladores experimentados. Además, es de código abierto, lo que significa que su código fuente está disponible para que cualquiera lo estudie, modifique y distribuya libremente.

PYTHON COMO LENGUAJE DE PROGRAMACIÓN:

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general este tiene:

Sintaxis legible y clara: Python se caracteriza por tener una sintaxis sencilla y legible, lo que facilita la comprensión del código y la escritura de programas más mantenibles.
Multiparadigma: Python soporta varios paradigmas de programación, incluyendo la programación orientada a objetos, la programación imperativa y la programación funcional, lo que permite a los desarrolladores elegir el enfoque más adecuado para sus necesidades.
Amplia biblioteca estándar: Python viene con una biblioteca estándar extensa que abarca áreas como manipulación de archivos, procesamiento de texto, networking, bases de datos, y más. Esto permite a los desarrolladores acceder a una amplia gama de funcionalidades sin tener que instalar bibliotecas adicionales.
Portabilidad: Python es multiplataforma, lo que significa que los programas escritos en Python pueden ejecutarse en diversos sistemas operativos como Windows, macOS y Linux sin necesidad de modificaciones significativas.
Comunidad activa: Python cuenta con una gran y activa comunidad de desarrolladores que contribuyen con bibliotecas, frameworks y herramientas, lo que hace que sea fácil encontrar ayuda, documentación y recursos en línea.
Aplicaciones diversas: Python se utiliza en una amplia variedad de campos, incluyendo desarrollo web, análisis de datos, inteligencia artificial, aprendizaje automático, scripting, automatización, desarrollo de juegos, y más (Lucas, 2019) [1].

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

VISUAL ESTUDIO

Visual Studio Code (también conocido como VS Code) es un editor de código fuente desarrollado por Microsoft que es muy popular entre los desarrolladores de software. Aunque comparte el nombre "Visual Studio" con el entorno de desarrollo integrado (IDE) más completo de Microsoft, Visual Studio Code es una herramienta más ligera y orientada principalmente al desarrollo de aplicaciones web y de escritorio, aunque también es utilizado para una variedad de otros propósitos.

Aquí hay algunas características destacadas de Visual Studio Code:

Editor de texto avanzado: VS Code ofrece un editor de texto altamente personalizable con resaltado de sintaxis, autocompletado inteligente, sugerencias de código, fragmentos de código (snippets), entre otras características.

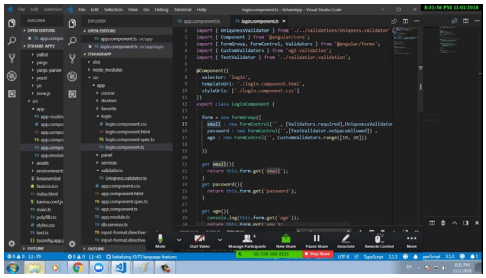
Soporte para múltiples lenguajes de programación: Es compatible con una amplia variedad de lenguajes de programación, gracias a su arquitectura extensible que permite la instalación de extensiones para añadir soporte a diferentes tecnologías y frameworks.

Integración con Git: Viene con integración nativa con Git, lo que facilita el control de versiones y la colaboración en proyectos de desarrollo de software.

Depuración integrada: Permite depurar aplicaciones directamente desde el editor, con soporte para puntos de interrupción, seguimiento de variables y más.

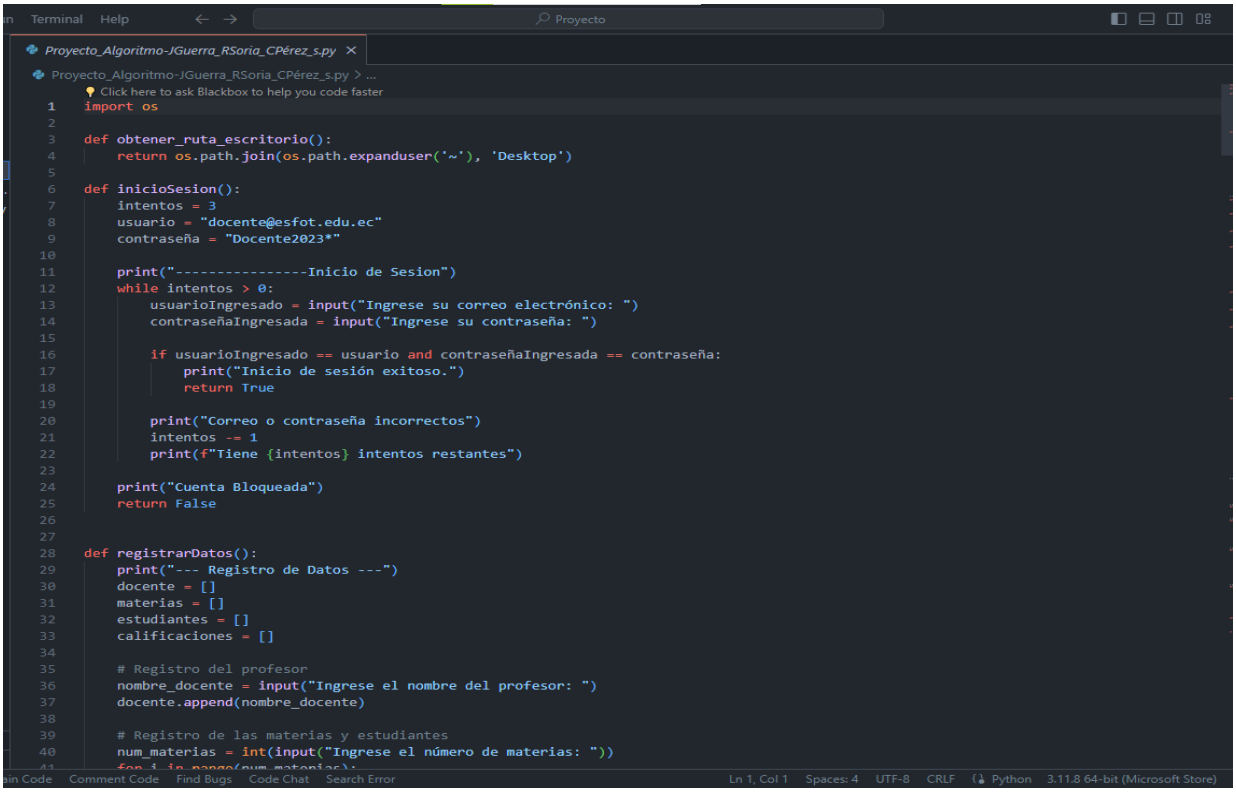
Terminal integrada: Incluye una terminal integrada que permite ejecutar comandos y scripts directamente desde el editor, sin necesidad de cambiar a una ventana de terminal separada.

Extensibilidad: Visual Studio Code es altamente extensible a través de su ecosistema de extensiones. Hay miles de extensiones disponibles que agregan nuevas funcionalidades, integraciones con servicios en la nube, herramientas de desarrollo específicas y mucho más [1].



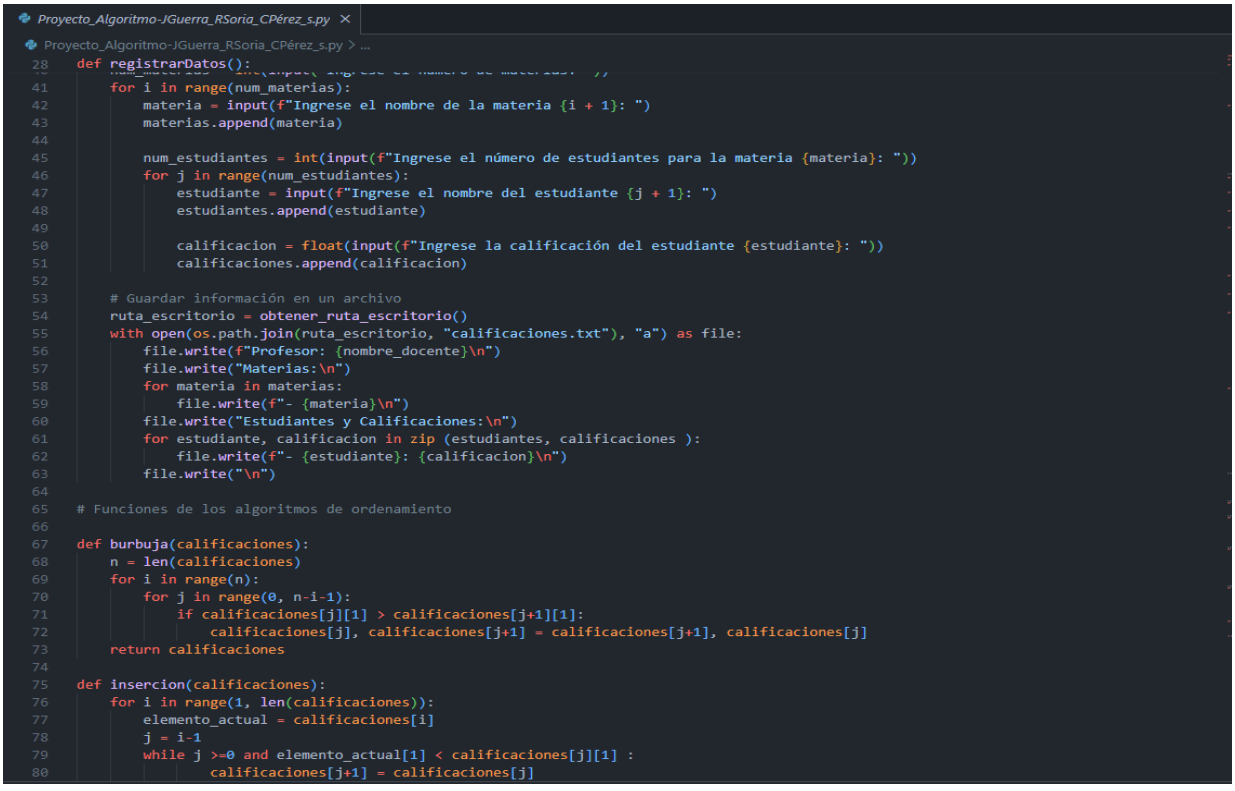
5. PROGRAMA

Figura 1:



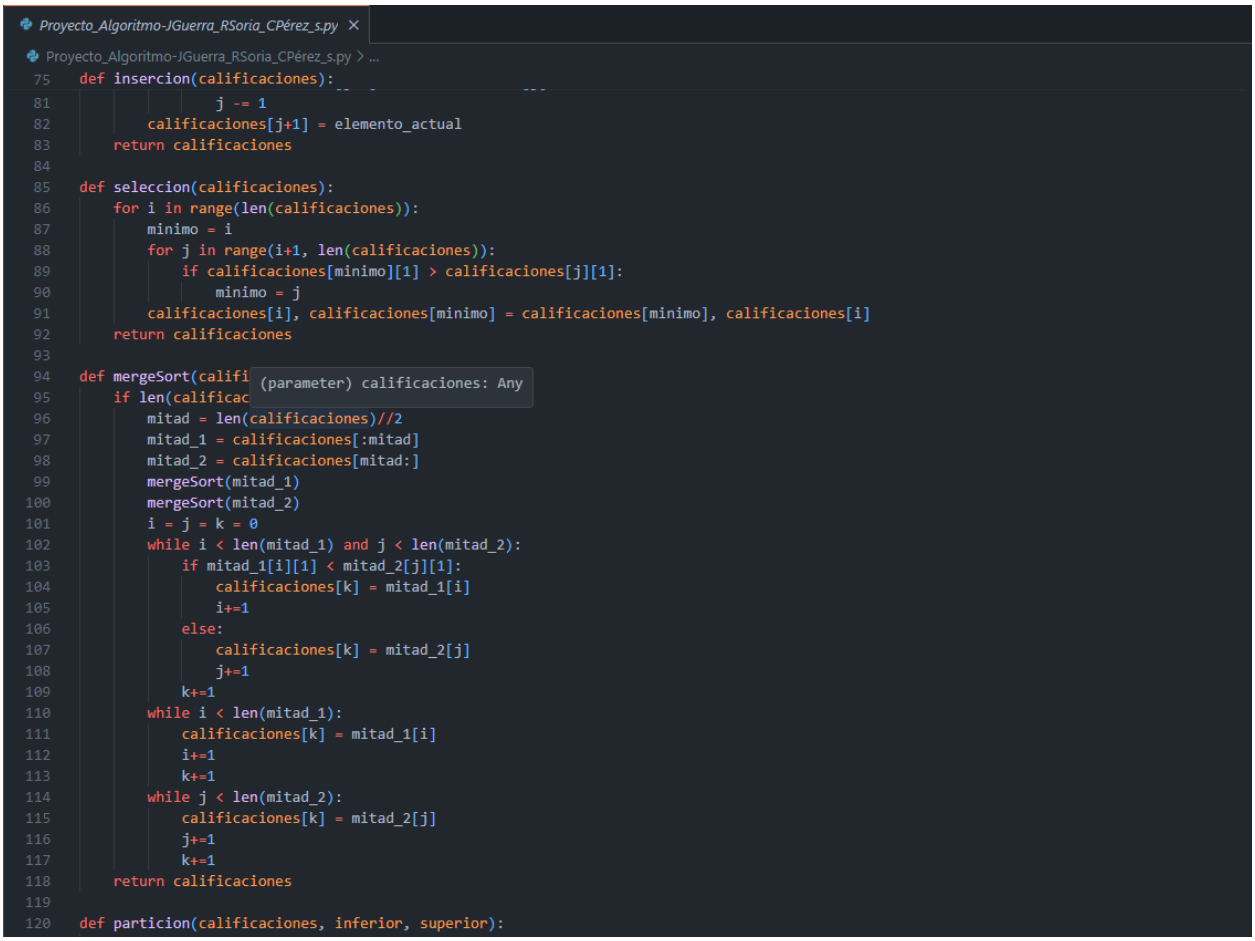
```python
import os

def obtener_ruta_escritorio():
    return os.path.join(os.path.expanduser('~'), 'Desktop')

def inicioSesion():
    intentos = 3
    usuario = "docente@esfot.edu.ec"
    contraseña = "Docente2023*"

    print("----------------Inicio de Sesion")
    while intentos > 0:
        usuarioIngresado = input("Ingrese su correo electrónico: ")
        contraseñaIngresada = input("Ingrese su contraseña: ")

        if usuarioIngresado == usuario and contraseñaIngresada == contraseña:
            print("Inicio de sesión exitoso.")
            return True

        print("Correo o contraseña incorrectos")
        intentos -= 1
        print(f"Tiene {intentos} intentos restantes")

    print("Cuenta Bloqueada")
    return False


def registrarDatos():
    print("--- Registro de Datos ---")
    docente = []
    materias = []
    estudiantes = []
    calificaciones = []

    # Registro del profesor
    nombre_docente = input("Ingrese el nombre del profesor: ")
    docente.append(nombre_docente)

    # Registro de las materias y estudiantes
    num_materias = int(input("Ingrese el número de materias: "))
```

Figura 2:



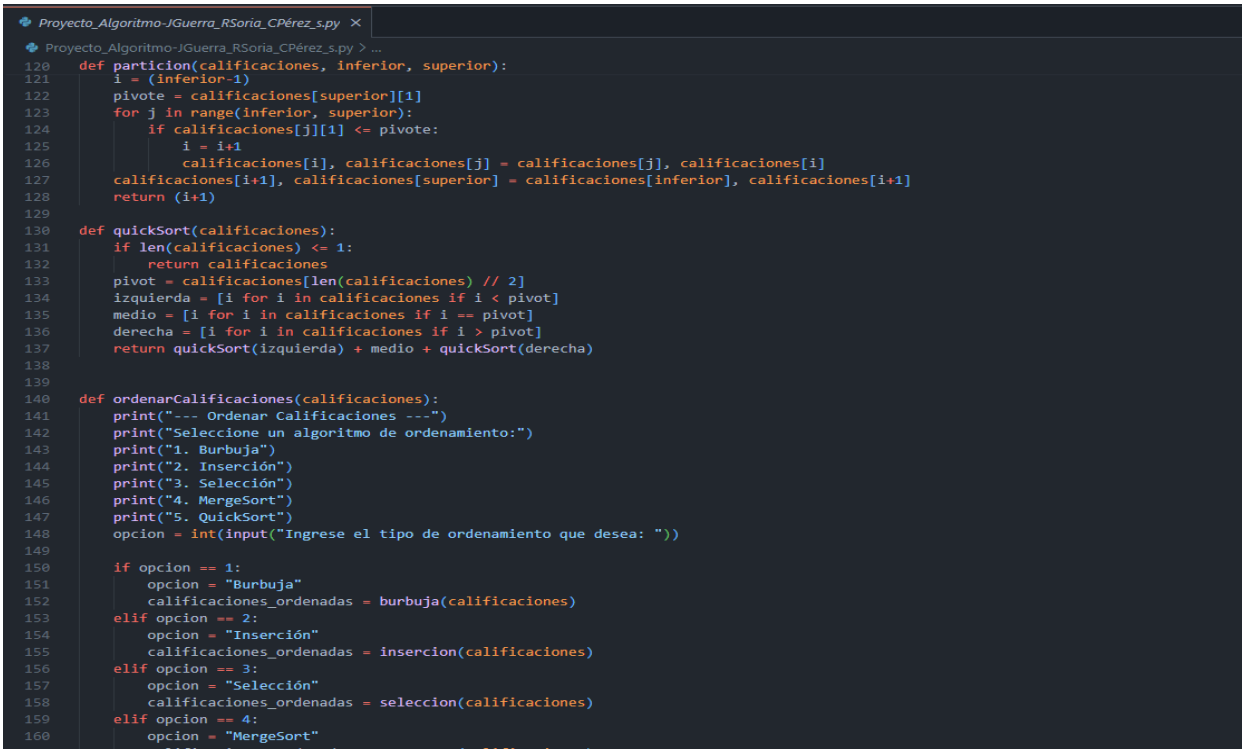
```python
def registrarDatos():
    for i in range(num_materias):
        materia = input(f"Ingrese el nombre de la materia {i + 1}: ")
        materias.append(materia)

        num_estudiantes = int(input(f"Ingrese el número de estudiantes para la materia {materia}: "))
        for j in range(num_estudiantes):
            estudiante = input(f"Ingrese el nombre del estudiante {j + 1}: ")
            estudiantes.append(estudiante)

            calificacion = float(input(f"Ingrese la calificación del estudiante {estudiante}: "))
            calificaciones.append(calificacion)

    # Guardar información en un archivo
    ruta_escritorio = obtener_ruta_escritorio()
    with open(os.path.join(ruta_escritorio, "calificaciones.txt"), "a") as file:
        file.write(f"Profesor: {nombre_docente}\n")
        file.write("Materias:\n")
        for materia in materias:
            file.write(f"- {materia}\n")
        file.write("Estudiantes y Calificaciones:\n")
        for estudiante, calificacion in zip (estudiantes, calificaciones ):
            file.write(f"- {estudiante}: {calificacion}\n")
        file.write("\n")

# Funciones de los algoritmos de ordenamiento

def burbuja(calificaciones):
    n = len(calificaciones)
    for i in range(n):
        for j in range(0, n-i-1):
            if calificaciones[j][1] > calificaciones[j+1][1]:
                calificaciones[j], calificaciones[j+1] = calificaciones[j+1], calificaciones[j]
    return calificaciones

def insercion(calificaciones):
    for i in range(1, len(calificaciones)):
        elemento_actual = calificaciones[i]
        j = i-1
        while j >=0 and elemento_actual[1] < calificaciones[j][1] :
                calificaciones[j+1] = calificaciones[j]
```

Figura 3:



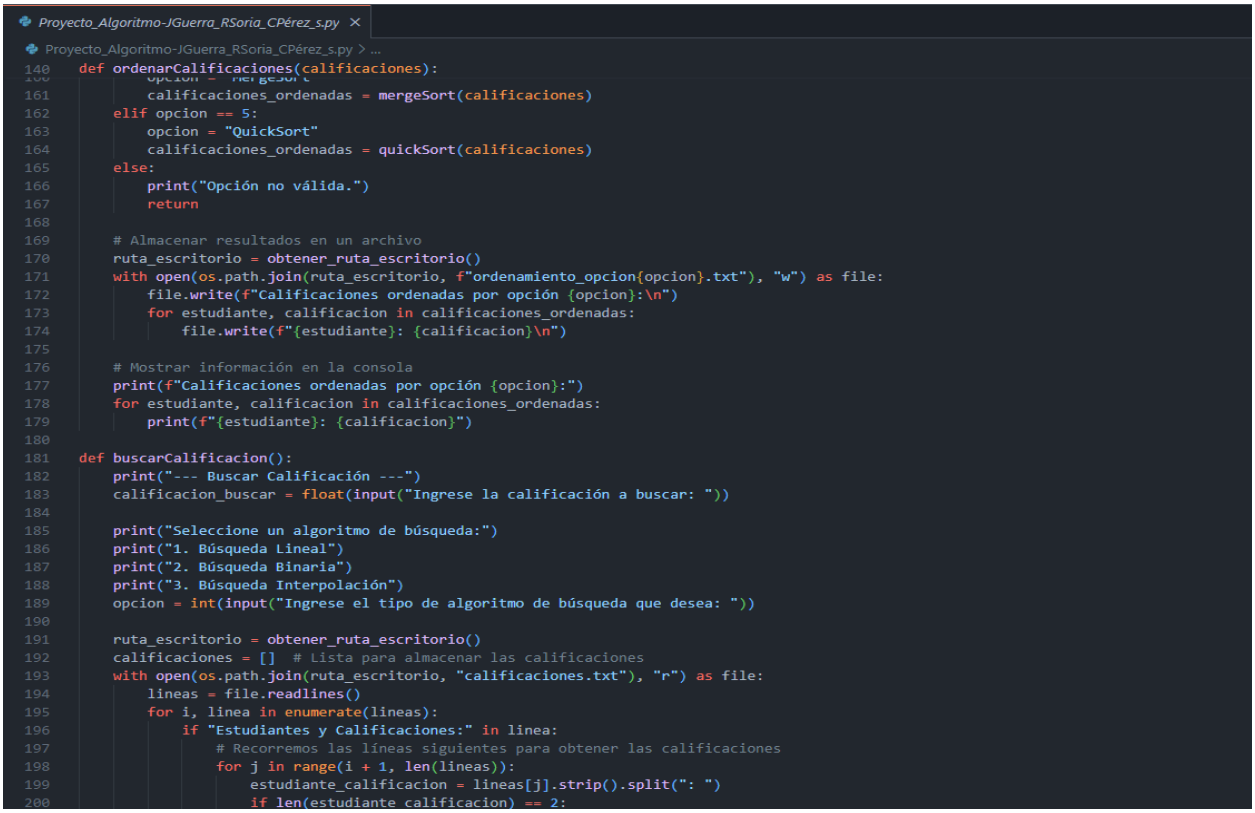
```
Proyecto_Algoritmo-JGuerra_RSoria_CPérez_s.py
Proyecto_Algoritmo-JGuerra_RSoria_CPérez_s.py > ...
75  def insercion(calificaciones):
81                  j -= 1
82          calificaciones[j+1] = elemento_actual
83      return calificaciones
84
85  def seleccion(calificaciones):
86      for i in range(len(calificaciones)):
87          minimo = i
88          for j in range(i+1, len(calificaciones)):
89              if calificaciones[minimo][1] > calificaciones[j][1]:
90                  minimo = j
91          calificaciones[i], calificaciones[minimo] = calificaciones[minimo], calificaciones[i]
92      return calificaciones
93
94  def mergeSort(calif(parameter) calificaciones: Any
95      if len(calificac
96          mitad = len(calificaciones)//2
97          mitad_1 = calificaciones[:mitad]
98          mitad_2 = calificaciones[mitad:]
99          mergeSort(mitad_1)
100         mergeSort(mitad_2)
101         i = j = k = 0
102         while i < len(mitad_1) and j < len(mitad_2):
103             if mitad_1[i][1] < mitad_2[j][1]:
104                 calificaciones[k] = mitad_1[i]
105                 i+=1
106             else:
107                 calificaciones[k] = mitad_2[j]
108                 j+=1
109             k+=1
110         while i < len(mitad_1):
111             calificaciones[k] = mitad_1[i]
112             i+=1
113             k+=1
114         while j < len(mitad_2):
115             calificaciones[k] = mitad_2[j]
116             j+=1
117             k+=1
118     return calificaciones
119
120 def particion(calificaciones, inferior, superior):
```

Figura 4:



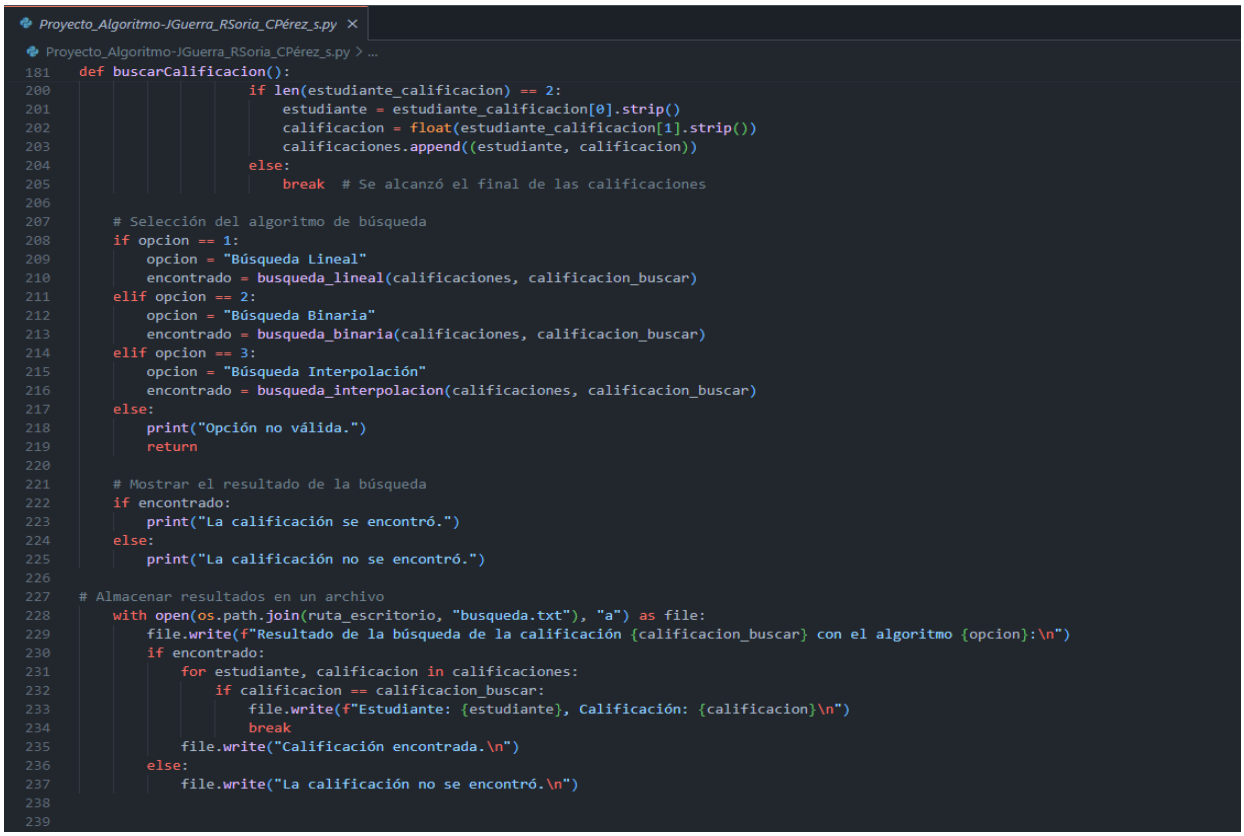
```
Proyecto_Algoritmo-JGuerra_RSoria_CPérez_s.py
Proyecto_Algoritmo-JGuerra_RSoria_CPérez_s.py > ...
120 def particion(calificaciones, inferior, superior):
121     i = (inferior-1)
122     pivote = calificaciones[superior][1]
123     for j in range(inferior, superior):
124         if calificaciones[j][1] <= pivote:
125             i = i+1
126             calificaciones[i], calificaciones[j] = calificaciones[j], calificaciones[i]
127     calificaciones[i+1], calificaciones[superior] = calificaciones[inferior], calificaciones[i+1]
128     return (i+1)
129
130 def quickSort(calificaciones):
131     if len(calificaciones) <= 1:
132         return calificaciones
133     pivot = calificaciones[len(calificaciones) // 2]
134     izquierda = [i for i in calificaciones if i < pivot]
135     medio = [i for i in calificaciones if i == pivot]
136     derecha = [i for i in calificaciones if i > pivot]
137     return quickSort(izquierda) + medio + quickSort(derecha)
138
139
140 def ordenarCalificaciones(calificaciones):
141     print("--- Ordenar Calificaciones ---")
142     print("Seleccione un algoritmo de ordenamiento:")
143     print("1. Burbuja")
144     print("2. Inserción")
145     print("3. Selección")
146     print("4. MergeSort")
147     print("5. QuickSort")
148     opcion = int(input("Ingrese el tipo de ordenamiento que desea: "))
149
150     if opcion == 1:
151         opcion = "Burbuja"
152         calificaciones_ordenadas = burbuja(calificaciones)
153     elif opcion == 2:
154         opcion = "Inserción"
155         calificaciones_ordenadas = insercion(calificaciones)
156     elif opcion == 3:
157         opcion = "Selección"
158         calificaciones_ordenadas = seleccion(calificaciones)
159     elif opcion == 4:
160         opcion = "MergeSort"
```

Figura 5:



```python
def ordenarCalificaciones(calificaciones):
        calificaciones_ordenadas = mergeSort(calificaciones)
    elif opcion == 5:
        opcion = "QuickSort"
        calificaciones_ordenadas = quickSort(calificaciones)
    else:
        print("Opción no válida.")
        return

    # Almacenar resultados en un archivo
    ruta_escritorio = obtener_ruta_escritorio()
    with open(os.path.join(ruta_escritorio, f"ordenamiento_opcion{opcion}.txt"), "w") as file:
        file.write(f"Calificaciones ordenadas por opción {opcion}:\n")
        for estudiante, calificacion in calificaciones_ordenadas:
            file.write(f"{estudiante}: {calificacion}\n")

    # Mostrar información en la consola
    print(f"Calificaciones ordenadas por opción {opcion}:")
    for estudiante, calificacion in calificaciones_ordenadas:
        print(f"{estudiante}: {calificacion}")

def buscarCalificacion():
    print("--- Buscar Calificación ---")
    calificacion_buscar = float(input("Ingrese la calificación a buscar: "))

    print("Seleccione un algoritmo de búsqueda:")
    print("1. Búsqueda Lineal")
    print("2. Búsqueda Binaria")
    print("3. Búsqueda Interpolación")
    opcion = int(input("Ingrese el tipo de algoritmo de búsqueda que desea: "))

    ruta_escritorio = obtener_ruta_escritorio()
    calificaciones = []  # Lista para almacenar las calificaciones
    with open(os.path.join(ruta_escritorio, "calificaciones.txt"), "r") as file:
        lineas = file.readlines()
        for i, linea in enumerate(lineas):
            if "Estudiantes y Calificaciones:" in linea:
                # Recorremos las líneas siguientes para obtener las calificaciones
                for j in range(i + 1, len(lineas)):
                    estudiante_calificacion = lineas[j].strip().split(": ")
                    if len(estudiante_calificacion) == 2:
```

Figura 6:



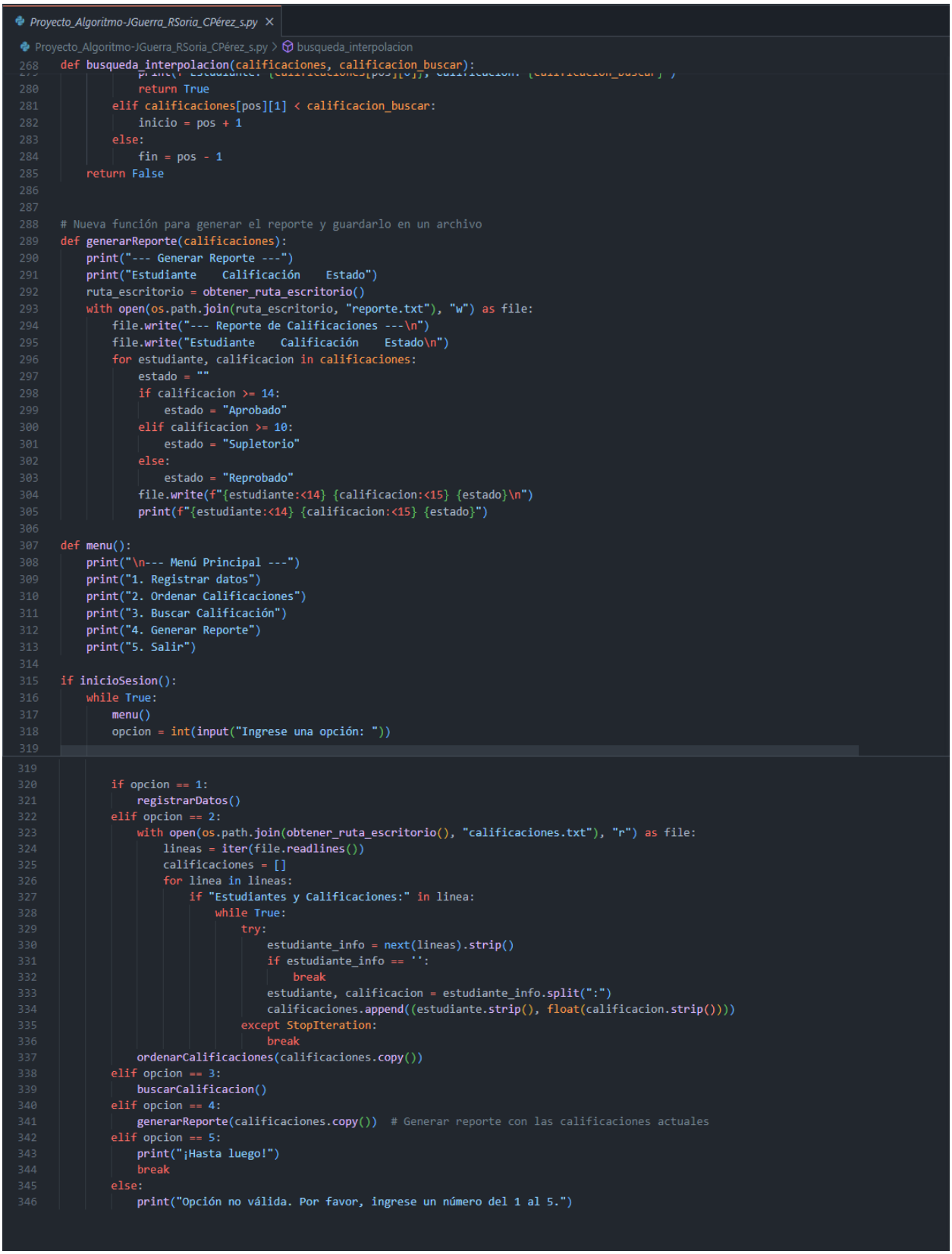
```python
def buscarCalificacion():
                    if len(estudiante_calificacion) == 2:
                        estudiante = estudiante_calificacion[0].strip()
                        calificacion = float(estudiante_calificacion[1].strip())
                        calificaciones.append((estudiante, calificacion))
                    else:
                        break  # Se alcanzó el final de las calificaciones

    # Selección del algoritmo de búsqueda
    if opcion == 1:
        opcion = "Búsqueda Lineal"
        encontrado = busqueda_lineal(calificaciones, calificacion_buscar)
    elif opcion == 2:
        opcion = "Búsqueda Binaria"
        encontrado = busqueda_binaria(calificaciones, calificacion_buscar)
    elif opcion == 3:
        opcion = "Búsqueda Interpolación"
        encontrado = busqueda_interpolacion(calificaciones, calificacion_buscar)
    else:
        print("Opción no válida.")
        return

    # Mostrar el resultado de la búsqueda
    if encontrado:
        print("La calificación se encontró.")
    else:
        print("La calificación no se encontró.")

# Almacenar resultados en un archivo
    with open(os.path.join(ruta_escritorio, "busqueda.txt"), "a") as file:
        file.write(f"Resultado de la búsqueda de la calificación {calificacion_buscar} con el algoritmo {opcion}:\n")
        if encontrado:
            for estudiante, calificacion in calificaciones:
                if calificacion == calificacion_buscar:
                    file.write(f"Estudiante: {estudiante}, Calificación: {calificacion}\n")
                    break
            file.write("Calificación encontrada.\n")
        else:
            file.write("La calificación no se encontró.\n")
```

Figura 7:

Proyecto_Algoritmo-JGuerra_RSoria_CPérez_s.py

```
240
241
242 # Algoritmo de búsqueda lineal
243 def busqueda_lineal(calificaciones, calificacion_buscar):
244     for estudiante, calificacion in calificaciones:
245         if calificacion == calificacion_buscar:
246             print(f"Estudiante: {estudiante}, Calificación: {calificacion}")
247             return True
248     return False
249
250 # Algoritmo de búsqueda binaria
251 def busqueda_binaria(calificaciones, calificacion_buscar):
252     calificaciones.sort(key=lambda x: x[1])  # Ordenar la lista de calificaciones por calificación
253     inicio = 0
254     fin = len(calificaciones) - 1
255
256     while inicio <= fin:
257         medio = (inicio + fin) // 2
258         if calificaciones[medio][1] == calificacion_buscar:
259             print(f"Estudiante: {calificaciones[medio][0]}, Calificación: {calificacion_buscar}")
260             return True
261         elif calificaciones[medio][1] < calificacion_buscar:
262             inicio = medio + 1
263         else:
264             fin = medio - 1
265     return False
266
267 # Algoritmo de búsqueda por interpolación
268 def busqueda_interpolacion(calificaciones, calificacion_buscar):
269     calificaciones.sort(key=lambda x: x[1])  # Ordenar la lista de calificaciones por calificación
270     n = len(calificaciones)
271     inicio = 0
272     fin = n - 1
273
274     while inicio <= fin and calificaciones[inicio][1] <= calificacion_buscar <= calificaciones[fin][1]:
275         # Interpolación para encontrar la posición más probable
276         pos = inicio + int((calificacion_buscar - calificaciones[inicio][1]) * (fin - inicio) / (calificaciones[fin][1] - calificaciones[
277
278         if calificaciones[pos][1] == calificacion_buscar:
279             print(f"Estudiante: {calificaciones[pos][0]}, Calificación: {calificacion_buscar}")
280             return True
```

Figura 8:



```python
def busqueda_interpolacion(calificaciones, calificacion_buscar):
            return True
        elif calificaciones[pos][1] < calificacion_buscar:
            inicio = pos + 1
        else:
            fin = pos - 1
    return False


# Nueva función para generar el reporte y guardarlo en un archivo
def generarReporte(calificaciones):
    print("--- Generar Reporte ---")
    print("Estudiante    Calificación    Estado")
    ruta_escritorio = obtener_ruta_escritorio()
    with open(os.path.join(ruta_escritorio, "reporte.txt"), "w") as file:
        file.write("--- Reporte de Calificaciones ---\n")
        file.write("Estudiante    Calificación    Estado\n")
        for estudiante, calificacion in calificaciones:
            estado = ""
            if calificacion >= 14:
                estado = "Aprobado"
            elif calificacion >= 10:
                estado = "Supletorio"
            else:
                estado = "Reprobado"
            file.write(f"{estudiante:<14} {calificacion:<15} {estado}\n")
            print(f"{estudiante:<14} {calificacion:<15} {estado}")

def menu():
    print("\n--- Menú Principal ---")
    print("1. Registrar datos")
    print("2. Ordenar Calificaciones")
    print("3. Buscar Calificación")
    print("4. Generar Reporte")
    print("5. Salir")

if inicioSesion():
    while True:
        menu()
        opcion = int(input("Ingrese una opción: "))

        if opcion == 1:
            registrarDatos()
        elif opcion == 2:
            with open(os.path.join(obtener_ruta_escritorio(), "calificaciones.txt"), "r") as file:
                lineas = iter(file.readlines())
                calificaciones = []
                for linea in lineas:
                    if "Estudiantes y Calificaciones:" in linea:
                        while True:
                            try:
                                estudiante_info = next(lineas).strip()
                                if estudiante_info == '':
                                    break
                                estudiante, calificacion = estudiante_info.split(":")
                                calificaciones.append((estudiante.strip(), float(calificacion.strip())))
                            except StopIteration:
                                break
            ordenarCalificaciones(calificaciones.copy())
        elif opcion == 3:
            buscarCalificacion()
        elif opcion == 4:
            generarReporte(calificaciones.copy())  # Generar reporte con las calificaciones actuales
        elif opcion == 5:
            print("¡Hasta luego!")
            break
        else:
            print("Opción no válida. Por favor, ingrese un número del 1 al 5.")
```

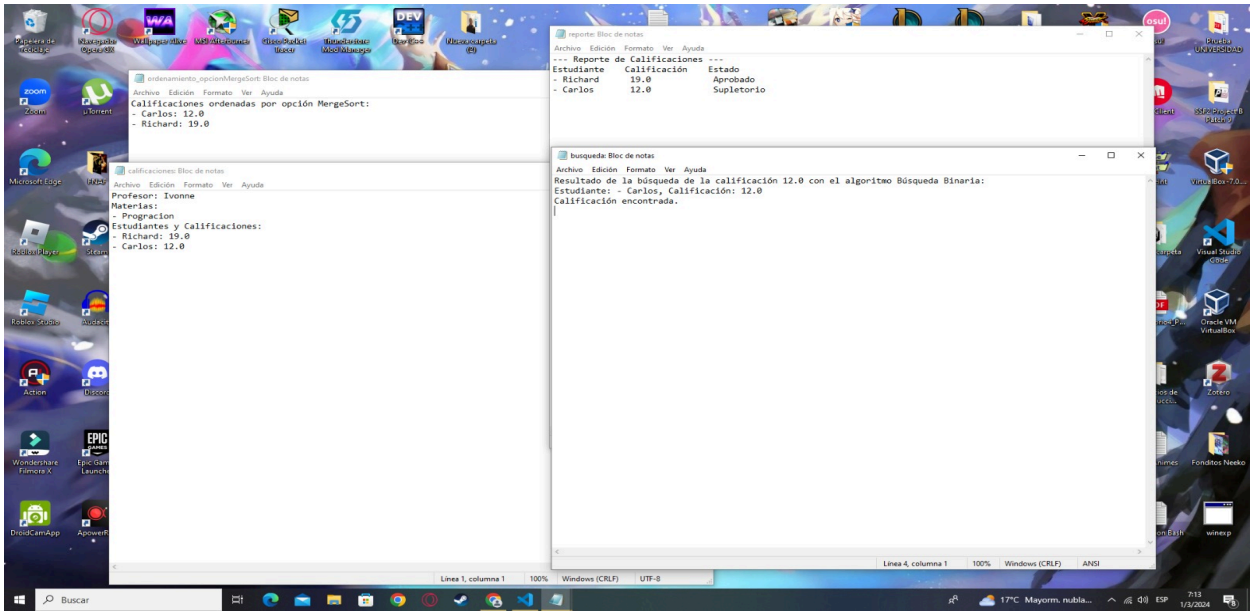
6. EJECUCIÓN DEL PROGRAMA:

Figura 1:

Figura 2:

```
--- Menú Principal ---
1. Registrar datos
2. Ordenar Calificaciones
3. Buscar Calificación
4. Generar Reporte
5. Salir
Ingrese una opción: 2
--- Ordenar Calificaciones ---
Seleccione un algoritmo de ordenamiento:
1. Burbuja
2. Inserción
3. Selección
4. MergeSort
5. QuickSort
Ingrese el tipo de ordenamiento que desea: 2
Calificaciones ordenadas por opción Inserción:
- Leonidas: 12.0
- Carlos: 19.0

--- Menú Principal ---
1. Registrar datos
2. Ordenar Calificaciones
3. Buscar Calificación
4. Generar Reporte
5. Salir
Ingrese una opción: 3
--- Buscar Calificación ---
Ingrese la calificación a buscar: 12
Seleccione un algoritmo de búsqueda:
1. Búsqueda Lineal
2. Búsqueda Binaria
3. Búsqueda Interpolación
Ingrese el tipo de algoritmo de búsqueda que desea: 3
Estudiante: - Leonidas, Calificación: 12.0
La calificación se encontró.

--- Menú Principal ---
1. Registrar datos
2. Ordenar Calificaciones
3. Buscar Calificación
4. Generar Reporte
5. Salir
Ingrese una opción: 4
--- Generar Reporte ---
Estudiante    Calificación    Estado
- Carlos       19.0            Aprobado
- Leonidas     12.0            Supletorio

--- Menú Principal ---
1. Registrar datos
2. Ordenar Calificaciones
3. Buscar Calificación
4. Generar Reporte
5. Salir
```

Figura 3:



EXTRAS

- Se realizo la investigación para la importación de una librería el cual busca la ruta de escritorio guardando todos los archivos.txt en el mismo y así evitar entrar a la carpeta C:\Users\ donde por lo general se guarda cuando se ejecuta tomado también en cuenta que no en todos los dispositivos será la misma ruta predeterminada por tanto se define que el programa busque la ruta de escritorio donde se ejecute el programa, y así guardara automáticamente, dicha librería tiene por nombre OS.

- IMPORT OS:

La librería `os` de Python es una herramienta integral que permite a los desarrolladores interactuar con el sistema operativo de manera programática. Con funciones para la manipulación de archivos y directorios, facilita tareas como la creación, eliminación, copia y movimiento de archivos. Además, ofrece acceso a información del sistema operativo, incluyendo el nombre del sistema, la ruta del directorio actual y las variables de entorno. La librería `os.path` complementa estas capacidades al proporcionar funciones para trabajar con rutas de archivos de forma portable, independientemente de la plataforma.

Por otro lado, la librería `os` permite ejecutar comandos del sistema operativo desde un programa Python utilizando la función `os.system()`, brindando a los desarrolladores la capacidad de automatizar tareas del sistema. Además, ofrece funciones para trabajar con procesos, como la creación de procesos secundarios y el manejo de señales del sistema, lo que facilita el control de la ejecución de programas desde Python. En resumen, la librería `os` es una herramienta esencial para interactuar con el sistema operativo desde Python, proporcionando una amplia gama de funciones para realizar tareas relacionadas con el sistema de archivos, el entorno del sistema y la ejecución de comandos del sistema [1].

7. ENLACES:

Enlace GitHub:
https://github.com/JosueGuerra2023B/Estructuras-Datos2023B/tree/master/Proyecto%20Final_JGuerra_RSoria_Cp%C3%A9rez

Enlace YouTube:
https://youtu.be/MhhUf3Lpcjc?si=L7I-RULt_IHJaQEs

8. Bibliografía

[1] J. Lucas, «Openwebinars,» 23 09 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-python/.

[2] VISUALESTUDIOCODE, «VISUALESTUDIOCODE,» 2024. [En línea]. Available: https://www.bing.com/search?q=visual&qs=n&form=QBRE&sp=-1&ghc=1&lq=0&pq=visual&sc=10-6&sk=&cvid=2CE9E1D360AA477A922EBF7A303EB2D1&ghsh=0&ghacc=0&ghpl=. [Último acceso: 26 02 2024].

[3] V. E. Code, «Visual Estudio Code,» 2020. [En línea]. Available: https://docs.python.org/3/library/os.html. [Último acceso: 26 01 2024].