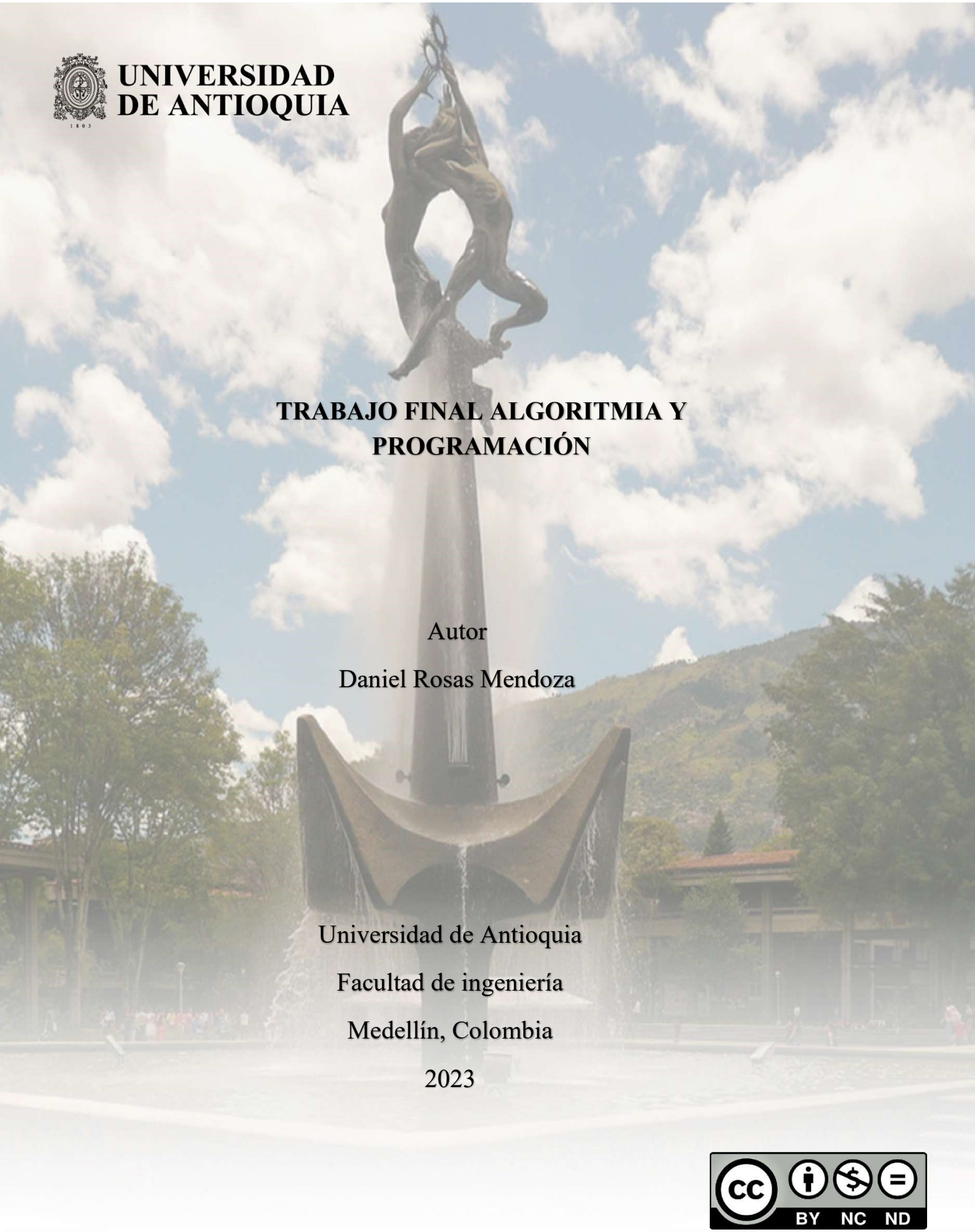

UNIVERSIDAD DE ANTIOQUIA

TRABAJO FINAL ALGORITMIA Y PROGRAMACIÓN

Autor

Daniel Rosas Mendoza

Universidad de Antioquia

Facultad de ingeniería

Medellín, Colombia

2023

CLASSDOC

Daniel Rosas Mendoza

Primera entrega presentada como requisito para el trabajo final

Profesor:
Julián Andrés Castillo Grisales

Línea de trabajo:
Estudio de problemas computacionales y soluciones algorítmicas

Universidad de Antioquia
Facultad de ingeniería
Medellín, Colombia
2023.

DOCUMENTO DE VISIÓN

PROYECTO CLASSDOC

Descripción General

La gestión eficaz de archivos y documentos se ha convertido en un desafío crucial tanto para las personas como para las organizaciones en la era actual, que en parte de debe a un aumento de la digitalización e informática en casi todas las esferas de la vida.
La demanda de una solución eficaz de gestión de archivos es mayor que nunca debido a la incesante acumulación de datos en diversos formatos, incluidos documentos de procesamiento de textos, hojas de cálculo, presentaciones y archivos multimedia.

En este contexto, el proyecto **“CLASSDOC”** es creado para ser una solución a esta creciente necesidad, un software integral y adaptable que permitiera a los usuarios organizar y categorizar automáticamente sus archivos, independientemente de su formato. Al proporcionar una herramienta potente y adaptable que simplifica drásticamente la gestión de archivos, **"CLASSDOC"** pretende revolucionar la forma en que las personas y las organizaciones gestionan sus activos digitales.

Objetivos y beneficios

Objetivo General

El objetivo general del proyecto **"CLASSDOC"** es desarrollar un software altamente eficiente, versátil y seguro que simplifique la organización y clasificación de archivos de múltiples formatos, garantizando una experiencia de usuario intuitiva y optimizando la gestión de documentos.

Objetivos Específicos

- **Mejorar la eficiencia y usabilidad:** Nuestro primer objetivo específico es diseñar y desarrollar un software que automatice la organización y clasificación de archivos, independientemente de la cantidad o variedad de formatos. Esto se logrará mediante la implementación de algoritmos avanzados y una interfaz de usuario intuitiva para que los usuarios, de todos los niveles de habilidad, puedan disfrutar de una experiencia eficiente y productiva al interactuar con el software.

- **Asegurar la versatilidad y seguridad:** El segundo objetivo específico se centra en la versatilidad y seguridad del software. Queremos que **"CLASSDOC"** sea altamente adaptable, capaz de trabajar con una amplia gama de formatos de archivos, incluyendo aquellos mencionados en la descripción del problema. Además, implementaremos medidas

de seguridad sólidas para garantizar la protección de los datos y la privacidad del usuario. Esto incluirá autenticación, cifrado y control de acceso adecuados.

- Optimizar la escalabilidad: El tercer objetivo específico es optimizar la escalabilidad del software. **"CLASSDOC"** deberá manejar sin problemas grandes volúmenes de archivos, manteniendo un rendimiento óptimo en todo momento. Esto se logrará a través de una arquitectura escalable y una gestión eficaz de recursos, asegurando que el software pueda crecer con las necesidades cambiantes de los usuarios sin comprometer la velocidad y la eficiencia.

Estos objetivos específicos se alinean con el objetivo general de desarrollar una solución de software completa y robusta que simplifique la gestión de archivos y brinde una experiencia de usuario de alta calidad.

Beneficios

-**Organización eficiente: "CLASSDOC"** permite a los usuarios manejar una gran cantidad de archivos en una variedad de formatos, desde documentos de oficina hasta imágenes. La solución reduce la carga de trabajo manual automatizando el proceso de organización.

- Clasificación inteligente: La capacidad de clasificación automática de "CLASSDOC" utiliza algoritmos sofisticados para identificar y categorizar archivos según su tipo. Esto garantiza que los archivos se almacenen en las carpetas correctas sin necesidad de manipularlos manualmente.

- Renombramiento uniforme: "CLASSDOC" establece un estándar de renombramiento uniforme para todos los archivos, lo que facilita la búsqueda y recuperación posterior. Para una mayor claridad y organización, los archivos se renombran de manera lógica y uniforme.

- Registro detallado: se registra detalladamente cada archivo procesado por **"CLASSDOC".** La información importante, como la ubicación del archivo anterior y actual, el tipo de archivo, el tamaño en unidades, así como la fecha de creación y modificación, se almacena en este archivo.

- Acceso Rápido: la organización y el registro minuciosos de los archivos permiten a los usuarios acceder a sus activos digitales rápidamente.

- Mejorar la productividad: "CLASSDOC" permite que los usuarios se centren en actividades más productivas y estratégicas al automatizar tareas de organización y clasificación que normalmente consumirían tiempo y esfuerzo.

ESPECIFICACIÓN DE REQUISITOS

REQUISITOS FUNCIONALES

- Clasificación automática: El software debe poder reconocer automáticamente el tipo de archivo según su formato y clasificarlo en la carpeta adecuada según su categoría.

- Renombramiento coherente: Todos los archivos deben tener un proceso de renombramiento coherente que garantice que cada archivo sea etiquetado de manera lógica y descriptiva para facilitar su búsqueda y recuperación.

- Registro detallado: El software debe registrar todos los detalles de cada archivo procesado, como su ruta anterior y nueva, el tipo de archivo, número de consonantes, el tamaño en diferentes unidades, la fecha de creación, la fecha de modificación, etc.

- Adaptabilidad a varios formatos: debe poder trabajar con muchos formatos comunes como Excel, Word, TXT, CSV, JSON, XML, PowerPoint, HTML y PNG.

REQUISITOS NO FUNCIONALES

-Rendimiento óptimo: El rendimiento de un software debe ser óptimo incluso en condiciones de carga de trabajo intensa, siendo capaz de manejar grandes volúmenes de archivos con facilidad.

-**Interfaz intuitiva:** La interfaz de usuario debe ser sencilla e intuitiva para que los usuarios de todos los niveles de experiencia puedan interactuar con el software con facilidad.

- Escalabilidad: El sistema debe ser escalable y expandirse para satisfacer las necesidades cambiantes de los usuarios sin afectar el rendimiento.

-**Compatibilidad:** Para garantizar una accesibilidad generalizada, se deben admitir sistemas operativos populares como Windows, macOS y Linux.

PLAN DEL PROYECTO

Plan de Actividades y cronograma para el proyecto "CLASSDOC" (5 semanas)

Semana 1: Análisis y diseño (1 semana):

- Dedicar la primera semana a un análisis rápido pero detallado de los requisitos.
- Diseñar una arquitectura y una estructura de datos simples y eficientes.
- Establecer un plan básico de implementación para identificar archivos y realizar el renombrado.

Semana 2-3: Desarrollo del software (2 semanas):

- Comenzar la implementación del software, priorizando las funciones esenciales.
- Centrarse en la lógica principal para clasificar archivos por tipo y realizar el renombrado básico.
- Empezar a trabajar en la funcionalidad para generar el Pandas Dataframe y registrar eventos en el log.

Semana 4: Pruebas y solución de errores (1 semana):

- Realizar pruebas rigurosas del software, enfocándose en las áreas críticas.
- Identificar y solucionar errores y problemas de manera eficiente.

Semana 5: Entrega (1 semana):

- Preparar el paquete de entrega, asegurando que el software esté listo y funcione de manera adecuada.
- Presentar el proyecto al profesor del curso, para asesorarse.
- Realizar una revisión final antes de la entrega para asegurarse de que todo esté completo y funcione correctamente.

PRESUPUESTO

Presupuesto para ejecutar el proyecto

Duración del trabajo: 5 semanas (25 días laborables)
Horas de trabajo diarias: 4 horas
Tarifa por hora del ingeniero: (1.160.000 COP / 30 días / 8 horas) * 4 horas = 4,613.33 COP/hora

Cálculo del presupuesto por semana:

- **Salario semanal =** (Horas diarias) x (tarifa por hora) x (días laborables por semana)
- **Salario semanal =** 4 horas/día x 4,613.33 COP/hora x 5 días/semana = 92,266.60 COP/semana

Presupuesto total para el proyecto (5 Semanas):

El presupuesto total para el ingeniero a cargo durante las 5 semanas de trabajo será de:

- **Presupuesto semanal x 5 semanas** = 92,266.60 COP/semana x 5 semanas = 461,333.00 COP

Presupuesto total

461,333.00 COP

PLAN DEL VERSIONADO "CLASSDOC"

El plan de versionado tiene como objetivo gestionar el desarrollo y lanzamiento de versiones del software "CLASSDOC" de manera estructurada y eficiente.
Esto incluye registrar cambios, numerar versiones y administrar ramas de desarrollo.

Política de numeración de versiones:
El software "CLASSDOC" seguirá una numeración de versión en el formato "X.Y.Z", donde:

"X" representa la versión principal.
"Y" representa la versión secundaria.
"Z" representa la versión de parche o revisión.

Estado Inicial - Versión 0.0.0

En la etapa inicial del proyecto **"CLASSDOC",** nos encontramos en la versión 0.0.0, que representa el punto de partida de nuestro desarrollo. En esta fase, no hemos comenzado aún la implementación del software en sí, pero hemos establecido las bases fundamentales para avanzar de manera sólida.

Logros iniciales o punto de partida:

- Hemos concebido y trazado el enfoque que se tomará para abordar el desafío de organizar y clasificar archivos en diversos formatos.

- Se ha aplicado un algoritmo que nos permite generar automáticamente 1000 archivos de diferentes tipos y extensiones, como Excel, Word, TXT, CSV, JSON, XML, PowerPoint, HTML y PNG.

- Se ha establecido una estructura para la numeración de versiones, siguiendo el formato "X.Y.Z," que nos guiará a lo largo del proyecto.

ALGORTIMO CLASSDOC

CÓDIGO CLASSDOC – ORGANIZADOR DE ARCHIVOS

```
# Importar las bibliotecas necesarias
import os  # Para operaciones del sistema operativo
import pandas as pd  # Para manipulación y análisis de datos
import shutil  # Para operaciones de archivos y carpetas
import datetime  # Para manejar fechas y tiempos
import sys # Para importar el sistema
# Configurar opciones de visualización de Pandas
pd.set_option('display.max_columns', None)  # Mostrar todas las
columnas en la salida
pd.set_option('display.max_rows', None)  # Mostrar todas las filas en
la salida
pd.set_option('display.width', None)  # No truncar la salida en ancho
pd.set_option('display.max_colwidth', None)  # No truncar el contenido
de las celdas

# Definir los directorios de entrada y salida
directorio_raiz = 'D:/UNIVERSIDAD 2023-2/ALGORITMOS Y
PROGRAMACION/ENTREGA 2/TrabajoFinal/tmp'
directorio_salida = 'D:/UNIVERSIDAD 2023-2/ALGORITMOS Y
PROGRAMACION/ENTREGA 2/TrabajoFinal/orden'

# Función para obtener el tipo de archivo a partir del nombre
def obtener_tipo_archivo(archivo):
    # Divide el nombre del archivo y su extensión
    nombre, extension = os.path.splitext(archivo)
    return extension

# Función para contar las vocales en el nombre del archivo
def contar_vocales(nombre):
    # Definir las vocales en mayúsculas y minúsculas
    vocales = "AEIOUaeiou"
    # Contar el número de vocales en el nombre
```

```
    return sum(1 for letra in nombre if letra in vocales)

# Función para contar las consonantes en el nombre del archivo
def contar_consonantes(nombre):
    # Definir las consonantes en mayúsculas y minúsculas
    consonantes = "BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz"
    # Contar el número de consonantes en el nombre
    return sum(1 for letra in nombre if letra in consonantes)

# Crear un DataFrame para almacenar la información de cada archivo
columnas = ['Nombre Anterior', 'Nombre Actual', 'Ruta Anterior', 'Nueva
Ruta', 'Tipo de Archivo', 'Tamaño (bit)', 'Tamaño (byte)', 'Tamaño
(Kilobyte)', 'Tamaño (Megabyte)', 'Vocales', 'Consonantes', 'Fecha de
Creación', 'Última Fecha de Modificación', 'Cantidad de Archivos del
Mismo Tipo']
df = pd.DataFrame(columns=columnas)

# Registro de eventos para mantener un seguimiento de las acciones
realizadas
log = []
log.append(f'Daniel Rosas , sistema operativo {os.name}, plataforma
{sys.platform}, fecha
{datetime.datetime.now().strftime("%Y%m%d%H%M%S")}')
log.append('"CLASSDOC" pretende revolucionar la forma en que las
personas y las organizaciones gestionan sus activos digitales.')
def registrar_evento(evento):
    # Registrar el evento con la marca de tiempo y el tiempo
transcurrido
    ahora = datetime.datetime.now()
    log.append(f"{ahora}\t{evento}\tTiempo empleado:
{str(datetime.datetime.now() - inicio)}")

# Iniciar el tiempo
inicio = datetime.datetime.now()

# Crear directorios para los tipos de archivos
tipos_de_archivo = set()

# Obtener todos los tipos de archivos presentes en el directorio raíz
for raiz, directorios, archivos in os.walk(directorio_raiz):
    for archivo in archivos:
        ruta_completa = os.path.join(raiz, archivo)
        tipo_archivo = obtener_tipo_archivo(archivo)
        tipos_de_archivo.add(tipo_archivo)

# Crear directorios para organizar los archivos según su tipo
for tipo in tipos_de_archivo:
    directorio_tipo = os.path.join(directorio_salida, tipo.strip('.'))

    # Cambiar nombres de carpetas según la extensión
    if tipo == '.xlsx':
        directorio_tipo = os.path.join(directorio_salida, 'excel')
    elif tipo == '.docx':
        directorio_tipo = os.path.join(directorio_salida, 'word')
    elif tipo == '.pptx':
        directorio_tipo = os.path.join(directorio_salida, 'powerpoint')
```

```
    os.makedirs(directorio_tipo, exist_ok=True)  # Crear directorios si
no existen

# Recorrer nuevamente el árbol de directorios y organizar los archivos
for raiz, directorios, archivos in os.walk(directorio_raiz):
    for archivo in archivos:
        ruta_completa = os.path.join(raiz, archivo)
        tipo_archivo = obtener_tipo_archivo(archivo)
        directorio_tipo = os.path.join(directorio_salida,
tipo_archivo.strip('.'))

        # Cambiar nombres de carpetas según la extensión
        if tipo_archivo == '.xlsx':
            directorio_tipo = os.path.join(directorio_salida, 'excel')
        elif tipo_archivo == '.docx':
            directorio_tipo = os.path.join(directorio_salida, 'word')
        elif tipo_archivo == '.pptx':
            directorio_tipo = os.path.join(directorio_salida,
'powerpoint')

        # Copiar el archivo al directorio correspondiente y renombrarlo
        nuevo_nombre = f"{len(os.listdir(directorio_tipo)) + 1:03d}-
{archivo}"
        nueva_ruta = os.path.join(directorio_tipo, nuevo_nombre)
        shutil.copy(ruta_completa, nueva_ruta)

        # Estadísticas del archivo
        estatisticas_archivo = [archivo, nuevo_nombre, raiz,
nueva_ruta, tipo_archivo]
        tamaño = os.path.getsize(nueva_ruta)
        estatisticas_archivo.extend([tamaño, tamaño / 8, tamaño / 8 /
1024, tamaño / 8 / 1024 / 1024])
        estatisticas_archivo.append(contar_vocales(archivo))
        estatisticas_archivo.append(contar_consonantes(archivo))

estatisticas_archivo.append(datetime.datetime.fromtimestamp(os.path.get
ctime(nueva_ruta)))

estatisticas_archivo.append(datetime.datetime.fromtimestamp(os.path.get
mtime(nueva_ruta)))

        # Contar la cantidad de archivos del mismo tipo
        cantidad_archivos_mismo_tipo = len(os.listdir(directorio_tipo))
        estatisticas_archivo.append(cantidad_archivos_mismo_tipo)

        df.loc[len(df)] = estatisticas_archivo

        evento = f"Mover {archivo} a {nuevo_nombre}"
        registrar_evento(evento)

# Guardar el DataFrame en un archivo Excel en el directorio de salida
ruta_resultado_excel = os.path.join(directorio_salida,
'archivo_resultado.xlsx')
# Verificar si el archivo ya existe
if os.path.exists(ruta_resultado_excel):
    # Si existe, intenta eliminarlo
    try:
```

```
        os.remove(ruta_resultado_excel)
    except Exception as e:
        print(f"No se pudo eliminar el archivo existente: {e}")

# Guardar el DataFrame en el nuevo archivo Excel
df.to_excel(ruta_resultado_excel, index=False)

# Mostrar el DataFrame en la consola
print(df)

# Guardar el registro de eventos en un archivo de texto en el
directorio de salida
ruta_registro_eventos = os.path.join(directorio_salida,
'registro_eventos.txt')
with open(ruta_registro_eventos, 'w') as archivo_log:
    for evento in log:
        archivo_log.write(evento + '\n')

# Finalizar el tiempo y registrar evento
fin = datetime.datetime.now()
registrar_evento(f"Procedimiento completado. Total de procedimientos
realizados: {len(log)}")

print("Proceso completado")
```

LINK EN GITHUB

https://github.com/drosas1295/CLASSDOC

Daniel Rosas Mendoza

1.036.339.232