UNIVERSIDAD NACIONAL
GUILLERMO BROWN

Materia: Estructura de datos

Trabajo práctico integrador: Cliente de correo electrónico - Entrega Final.

Comisión: 3

Profesor: Angel Leonardo Bianco

Grupo 11.

Alumnos: Brizuela Carla, Vergara Lucas.

Introducción al proyecto

Este proyecto consiste en el desarrollo de un servidor de mensajería simple, implementado en python. El sistema permite gestionar mensajes a través de diferentes carpetas y subcarpetas. A su vez permite enviar y recibir correos normales o con carácter de urgente.

Objetivos

- Desarrollar un servidor de correo básico en python para gestionar mensajes, carpetas y prioridades mediante POO.
- Implementar clases que representen usuarios, carpetas, mensajes y al servidor.
- Diseñar un mecanismo para poder clasificar y mostrar mensajes normales y urgentes.
- Implementar el movimiento de mensajes entre carpetas manteniendo sus prioridad.
- Implementar la creación de carpetas en el caso que sea necesario.

La primera clase que creamos fue la clase mensaje y su método mostrar.

```python
class Mensaje:
    def __init__(self, remitente, destinatario, asunto, contenido):
        self.remitente = remitente
        self.destinatario = destinatario
        self.asunto = asunto
        self.contenido = contenido

    def mostrar(self):
        print(f"De: {self.remitente}")
        print(f"Para: {self.destinatario}")
        print(f"Asunto: {self.asunto}")
        print(f"Contenido: {self.contenido}")
```

Parámetros:

Remitente

Destinatario

Asunto

Contenido

Luego la clase Usuario, carpeta y Servidor

```python
class Usuario:
    def __init__(self, nombre, correo):
        self.nombre = nombre
        self.correo = correo
        self.bandeja_recibidos = []
        self.bandeja_enviados = []
```

```python
class Servidor:
    def __init__(self, usuarios):
        self.usuarios = []

    def agregar_usuario():
        usuario = Usuario(self.nombre, self.correo)
```

```python
class Carpeta:
    def __init__(self, bandeja_entrada, bandeja_salida):
        self.bandeja_entrada = bandeja_entrada
        self.bandeja_salidad = bandeja_salida
```

Creamos las interfaces de enviar y recibir mensaje

```python
# INTERFAZ: ENVIAR MENSAJE
def enviar_mensaje(self, destinatario, asunto, contenido):
    mensaje = Mensaje(self.correo, destinatario.correo, asunto, contenido)
    self.bandeja_enviados.append(mensaje)  # lo guardo en enviados
    destinatario.recibir_mensaje(mensaje)  # se lo envío al destinatario

# INTERFAZ: RECIBIR MENSAJE
def recibir_mensaje(self, mensaje):
    self.bandeja_recibidos.append(mensaje)
```

Creamos la interfaz de la bandeja de entrada y salida

```python
# INTERFAZ: LISTAR BANDA DE ENTRADA
def mostrar_bandeja_entrada(self):
    print(f"\nBandeja de entrada de {self.nombre}:")
    if not self.bandeja_recibidos:
        print("No hay mensajes.")
    for m in self.bandeja_recibidos:
        m.mostrar()

# INTERFAZ: LISTAR BANDA DE SALIDA
def mostrar_bandeja_salida(self):
    print(f"\nBandeja de salida de {self.nombre}:")
    if not self.bandeja_enviados:
        print("No hay mensajes enviados.")
    for m in self.bandeja_enviados:
        m.mostrar()
```

Cambiamos la clase Usuario

```python
class Usuario:
    def __init__(self, nombre, correo):
        self.nombre = nombre
        self.correo = correo
        self.carpetas = {
            "entrada": Carpeta("Bandeja de entrada"),
            "enviados": Carpeta("Bandeja de enviados")
        }

    def enviar_mensaje(self, destinatario, asunto, contenido):
        mensaje = Mensaje(self.correo, destinatario.correo, asunto, contenido)
        self.carpetas["enviados"].agregar_mensaje(mensaje)
        destinatario.recibir_mensaje(mensaje)

    def recibir_mensaje(self, mensaje):
        self.carpetas["entrada"].agregar_mensaje(mensaje)

    def mostrar_carpeta(self, tipo):
        if tipo in self.carpetas:
            self.carpetas[tipo].mostrar_mensajes()
        else:
            print(f"La carpeta {tipo} no existe")
```

Cambios en la clase Servidor

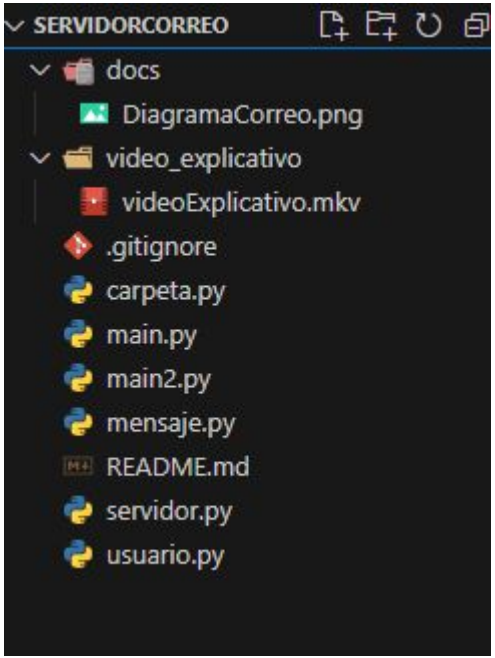
```python
class Servidor:
    def __init__(self):
        self.usuarios = []

    def agregar_usuario(self, usuario):
        self.usuarios.append(usuario)
        print(f"Se agregó el usuario: {usuario.nombre}")


    def mostrar_usuarios(self):
        for u in self.usuarios:
            print(f"{u.nombre} ({u.correo})")
```

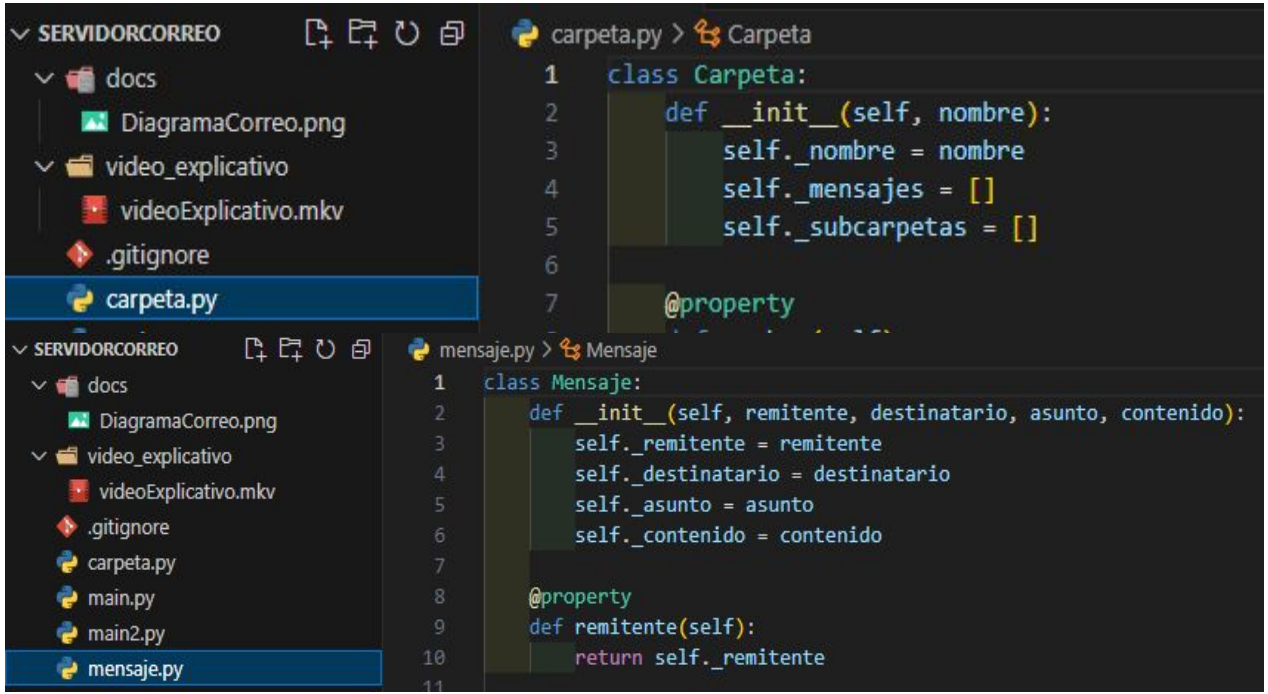
No teníamos un archivo MAIN.py

```python
#EJECUCIÓN DEL PROGRAMA

#*Creamos el servidor*
print()
servidor = Servidor()
print("El servidor de correo se ha sido inicializado")
print("-----------------------------------------")

#*Creamos los usuarios*
print("Los siguientes Usuarios se han agregado en el servidor:")
print()
lucas = Usuario("Lucas", "lucasvergara.f@gmail.com")
carla = Usuario("Carla", "carlabrizuela@gmail.com")


#*Agregamos los usuarios al servidor*
servidor.agregar_usuario(lucas)
servidor.agregar_usuario(carla)

#*Mostramos que los usuarios están en el servidor*

servidor.mostrar_usuarios()
print("--------------------")
```

```python
#*Creamos los usuarios*
print("Los siguientes Usuarios se han agregado en el servidor:")
print()
lucas = Usuario("Lucas", "lucasvergara.f@gmail.com")
carla = Usuario("Carla", "carlabrizuela@gmail.com")


#*Agregamos los usuarios al servidor*
servidor.agregar_usuario(lucas)
servidor.agregar_usuario(carla)

#*Mostramos que los usuarios están en el servidor*

servidor.mostrar_usuarios()
print("--------------------")

#*Enviamos un mensaje*

print(f"El usuario Lucas está enviado un mensaje")
print()
lucas.enviar_mensaje(carla, "Saludo", "Hola, Carla, ¿cómo estás?")
carla.enviar_mensaje(lucas, "Tanto tiempo!", "Hola Lucas, ¿cómo estás, tanto tiempo

#*Mostramos Bandeja de entrada del usuario Carla*

carla.mostrar_carpeta("entrada")
print()
print("Carla le responde el mensaje")
print()
lucas.mostrar_carpeta("entrada")
```

Modularización del proyecto

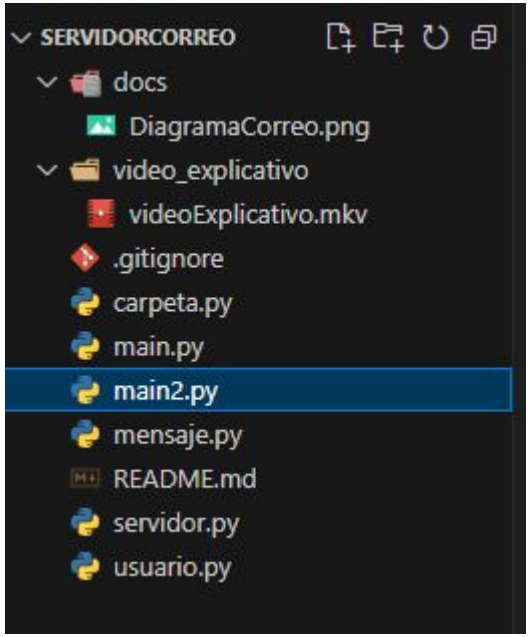




Encapsulamiento de clases

```python
class Mensaje:
    def __init__(self, remitente, destinatario, asunto, contenido):
        self._remitente = remitente
        self._destinatario = destinatario
        self._asunto = asunto
        self._contenido = contenido

    @property
    def remitente(self):
        return self._remitente

    @property
    def destinatario(self):
        return self._destinatario

    @property
    def asunto(self):
        return self._asunto

    @property
    def contenido(self):
        return self._contenido
```

Main con línea de comandos



main2.py

```python
from servidor import Servidor
from usuario import Usuario
from carpeta import Carpeta
from mensaje import Mensaje


def menu_servidor(servidor):
    while True:
        print("\n--- Servidor de correo ---")
        print("El servidor de correo cuenta con los siguientes usuarios:")
        servidor.mostrar_usuarios()
        print("\nOpciones:")
        print("1 - Enviar mensaje")
        print("2 - Buscar mensajes")
        print("3 - Mover mensajes")
        print("4 - Mostrar bandeja de entrada")
        print("5 - Mostrar bandeja de enviados")
        print("6 - Mostrar todas las carpetas del usuario")
        print("0 - Salir")

        opcion = input("Seleccione una opción: ").strip()
```

Buscar una nueva carpeta o subcarpeta mediante una función recursiva

```python
def buscar_mensajes(self, remitente=None, asunto=None, contenido=None, destinatario=None ):
    encontrados = []
    for m in self._mensajes:
        if (remitente is None or m.remitente == remitente) and \
        (asunto is None or m.asunto == asunto) \
            (contenido is None or m.contenido == contenido) \
            (destinatario is None or m.destinario == destinatario):
            encontrados.append(m)
    for sub in self._subcarpetas:
        encontrados.extend(sub.buscar_mensajes(remitente, asunto))
    return encontrados
```

Mover carpeta recursiva

```python
def mover_mensaje(self, mensaje, carpeta_destino):
    if mensaje in self._mensajes:
        self._mensajes.remove(mensaje)
        carpeta_destino.agregar_mensaje(mensaje)
        return True
    for sub in self._subcarpetas:
        if sub.mover_mensaje(mensaje, carpeta_destino):
            return True
    return False
```

(parameter) carpeta_destino: Any

Mostrar árbol completo

```python
def mostrar_arbol(self, nivel=0):
    print("    " * nivel + f"- {self.nombre}")
    for sub in self._subcarpetas:
        sub.mostrar_arbol(nivel + 1)
```