Cliente de Correo Electrónico

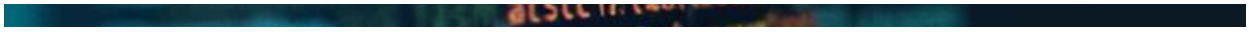
Universidad Nacional Guillermo Brown

Materia: Estructura de Datos

Entrega 1: Modelado de Clases y Encapsulamiento.

Fecha: 30/09

Integrantes: Ariel Aguilar, Ignacio Nicolás Heredia, Katherine Avendaño



INTRODUCCIÓN

El presente trabajo corresponde al Proyecto Final de la cátedra **Estructura de Datos**. El objetivo principal es diseñar e implementar, en el lenguaje de programación **Python,** un sistema orientado a objetos que modele un **cliente de Correo Electrónico**.

En esta primera entrega se abordan los conceptos de **modelado de clases y encapsulamiento**, con fin de construir la base de sistema. Se busca representar entidades fundamentales como **Usuarios, Mensajes, Carpetas y Servidores de correo**, utilizando los principios de la Programación Orientada a Objetos (POO).

Se incluyen ejemplos de cómo las clases interactúan entre sí y cómo se protege la información sensible mediante métodos públicos y propiedades, garantizando un sistema seguro y organizado desde sus primeras etapas.

OBJETIVOS

- Modelar las entidades principales de un cliente de correo electrónico mediante clases en Python.
- Aplicar el principio de **encapsulamiento** para proteger los atributos internos y controlar el acceso a los datos.
- Representar el diseño mediante un **Diagrama UML** que muestra las relaciones entre clases.
- Justificar las decisiones de diseño orientado a objetos en función de la claridad, modularidad y escalabilidad del sistema.
- Ilustrar con ejemplos el funcionamiento básico del sistema (envío de mensajes, organización en carpetas, conexión entre usuarios).

DISEÑO DE CLASES

Se identificaron las siguientes clases principales:

1. **Usuario:** representa a la persona que utiliza el sistema.

   > Atributos: nombre, dirección de correo, lista de carpetas.

   > Métodos: enviar mensajes, recibir mensajes, listar carpetas.

   Ejemplos:

   usuario1 = Usuario("Ana", "ana@mail.com")

   usuario2 = Usuario("Luis", "luis@mail.com")

usuario1.enviar_mensaje(usuario2, "Hola!", "¿Cómo estás?"

2. **Mensaje**: modela un correo electrónico.

    - Atributos: remitente, destinatario, asunto, cuerpo, fecha de envío.
    - Métodos: obtener información, mostrar resumen.

Ejemplo de encapsulamiento:

```
class Mensaje:
    def __init__(self, remitente, destinatario, asunto, cuerpo):
        self.__remitente = remitente
        self.__destinatario = destinatario
        self.__asunto = asunto
        self.__cuerpo = cuerpo

    @property
    def cuerpo(self):
        return self.__cuerpo

    @cuerpo.setter
    def cuerpo(self, nuevo_cuerpo):
        self.__cuerpo = nuevo_cuerpo
```

3. Carpeta: almacena y organiza mensajes.

    - Atributos: nombre de la carpeta, lista de mensajes.
    - Métodos: agregar mensaje, listar mensajes, buscar mensajes por remitente o asunto

Ejemplo:

```
bandeja = Carpeta("Bandeja de entrada")
bandeja.agregar_mensaje(mensaje1)
```

4. **ServidorCorreo:** administra múltiples usuarios y permite la conexión entre ellos.
    - Atributos: lista de usuarios.
    - Métodos: registrar usuario, enviar mensaje entre usuarios, buscar usuario por correo.

Ejemplo:

```
servidor = ServidorCorreo()

servidor.registrar_usuario(usuario1)

servidor.enviar_mensaje(usuario1, usuario2, "Hola!", "Primer mensaje desde el servidor")
```

ENCAPSULAMIENTO

El encapsulamiento se implementa para proteger la información sensible de cada clase.

- Los atributos de un mensaje (como cuerpo o destinatario) se definen como **privados o protegidos.**
- El acceso a estos atributos se realiza mediante **métodos públicos** (getters y setters) o propiedades en Python (@property).
- Esto evita modificaciones indebidas y asegura la **consistencia** del sistema.

Ejemplo de encapsulamiento en Usuario y Mensaje:

```
usuario1 = Usuario("Ana", "ana@mail.com")

print(usuario1.email)  # Acceso seguro mediante getter
```

JUSTIFICACIÓN DEL DISEÑO

El diseño propuesto refleja de forma clara las entidades y relaciones de un sistema de correo electrónico real:

- Se separan responsabilidades en **clases independientes** para garantizar **modularidad**.
- El encapsulamiento asegura **seguridad y control** sobre los datos internos.
- La orientación a objetos facilita futuras ampliaciones, como:
    - Filtros automáticos de mensajes
    - Organización en **árboles de carpetas**
    - Simulación de **red de servidores** mediante grafos

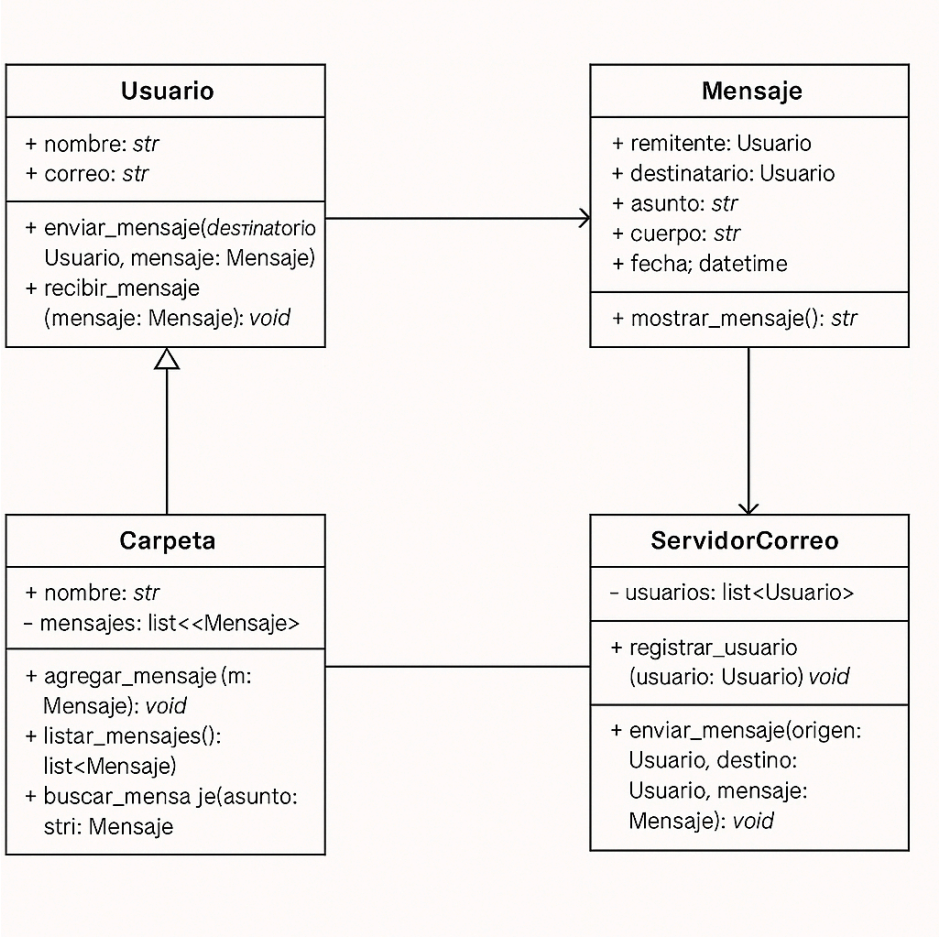
- Las relaciones entre clases permiten mantener un **flujo lógico de información,** reflejando cómo se comunicarán los usuarios dentro de un servidor de correo real.

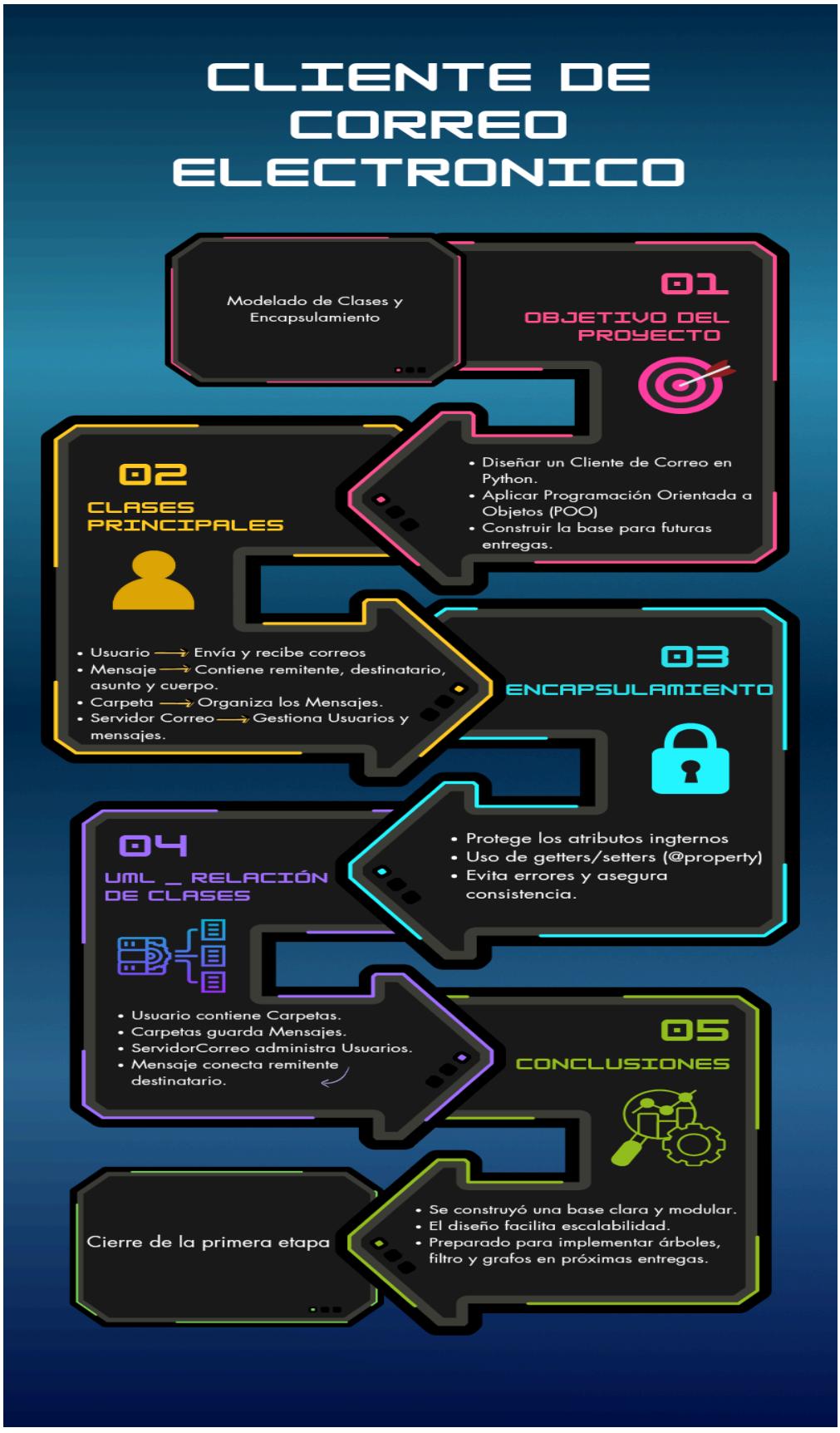
CONCLUSIONES

En esta primera entrega se establecieron las bases del cliente de correo electrónico mediante el modelado de clases y la aplicación del encapsulamiento.

- Se creó una estructura clara y modular para los objetos principales.
- Se garantiza la seguridad y consistencia de los datos mediante atributos **privados y métodos públicos.**
- El diseño permite agregar funcionalidades futuras sin comprometer la arquitectura inicial.

La claridad en el diseño inicial permitirá un desarrollo más eficiente, manteniendo **buenas prácticas de programación** y facilitando la comprensión del sistema en la defensa final.




CLIENTE DE CORREO ELECTRONICO
Modelado de Clases y Encapsulamiento
01
OBJETIVO DEL PROYECTO
• Diseñar un Cliente de Correo en Python.
• Aplicar Programación Orientada a Objetos (POO)
• Construir la base para futuras entregas.
02
CLASES PRINCIPALES
• Usuario → Envía y recibe correos
• Mensaje → Contiene remitente, destinatario, asunto y cuerpo.
• Carpeta → Organiza los Mensajes.
• Servidor Correo → Gestiona Usuarios y mensajes.
03
ENCAPSULAMIENTO
• Protege los atributos ingternos
• Uso de getters/setters (@property)
• Evita errores y asegura consistencia.
04
UML _ RELACIÓN DE CLASES
• Usuario contiene Carpetas.
• Carpetas guarda Mensajes.
• ServidorCorreo administra Usuarios.
• Mensaje conecta remitente destinatario.
05
CONCLUSIONES
• Se construyó una base clara y modular.
• El diseño facilita escalabilidad.
• Preparado para implementar árboles, filtro y grafos en próximas entregas.
Cierre de la primera etapa