Actividad 6.2 Ejercicio de programación 3 y pruebas de unidad

A01795690. Michelle Alejandro Durán Sánchez

Escuela de Ingeniería y Ciencias, Tecnológico de Monterrey

TC4017.10: Pruebas de software y aseguramiento de la calidad

Dr. Gerardo Padilla Zárate

Mtra. María Mylen Treviño E.

Ciudad de México, 16 de febrero de 2025

Ejercicio de programación

Ejercicio 1: Programming Exercise

Requerimiento:

Requisito 1: Implementar un conjunto de clases en Python que represente tres abstracciones:

1. Hotel
2. Reserva
3. Clientes

Requisito 2: Implementar un conjunto de métodos para gestionar los siguientes comportamientos persistentes (almacenados en archivos):

1. Hoteles
    a. Crear un hotel
    b. Eliminar un hotel
    c. Mostrar información del hotel
    d. Modificar información del hotel
    e. Reservar una habitación
    f. Cancelar una reserva
2. Clientes
    a. Crear un cliente
    b. Eliminar un cliente
    c. Mostrar información del cliente
    d. Modificar información del cliente
3. Reservas
    a. Crear una reserva (Cliente, Hotel)
    b. Cancelar una reserva

Se tiene libertad para definir los atributos dentro de cada clase que permitan el comportamiento requerido.

Requisito 3: Implementar casos de prueba unitarios para evaluar los métodos de cada clase. Usar el módulo "*unittest*" en Python.

Requisito 4: La cobertura de código de todas las pruebas unitarias debe alcanzar al menos un 85% de cobertura de líneas.

Requisito 5: El programa debe incluir un mecanismo para manejar datos inválidos en los archivos. Los errores deben mostrarse en la consola y la ejecución debe continuar.

Requisito 6: El código debe cumplir con PEP8.

Requisito 7: El código fuente no debe mostrar advertencias al ejecutarse con Flake8 y PyLint.

Resultados:

- Ejemplo si hubiera errores con PyLint:

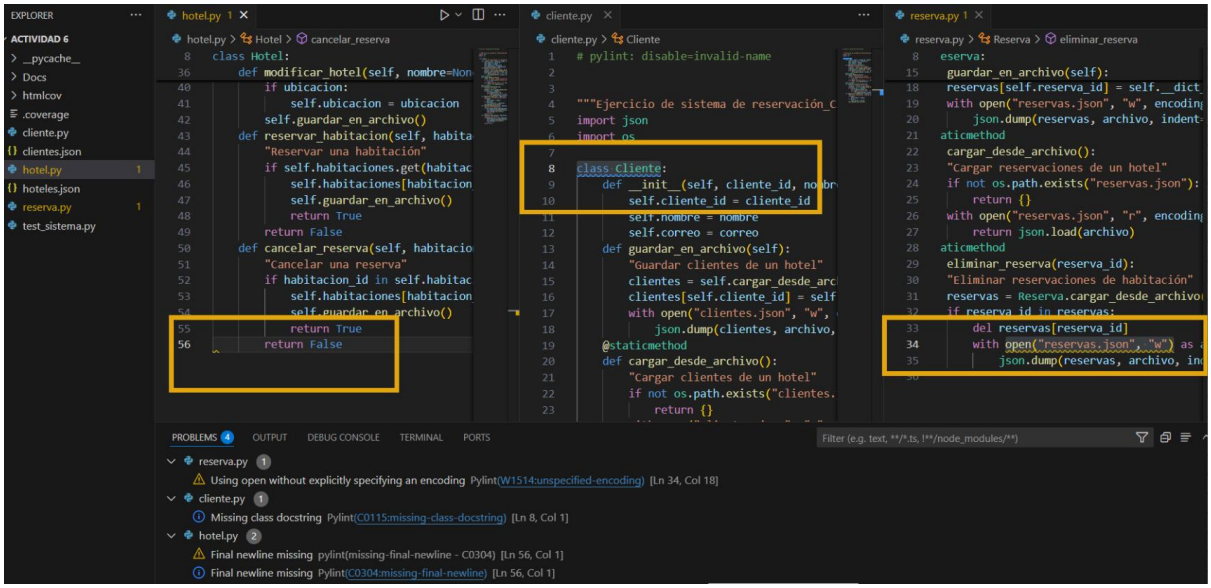

Imagen 1. Resultados con errores en PyLint de hotel.py, cliente.py, reserva.py.

- Archivos sin errores de PyLint:

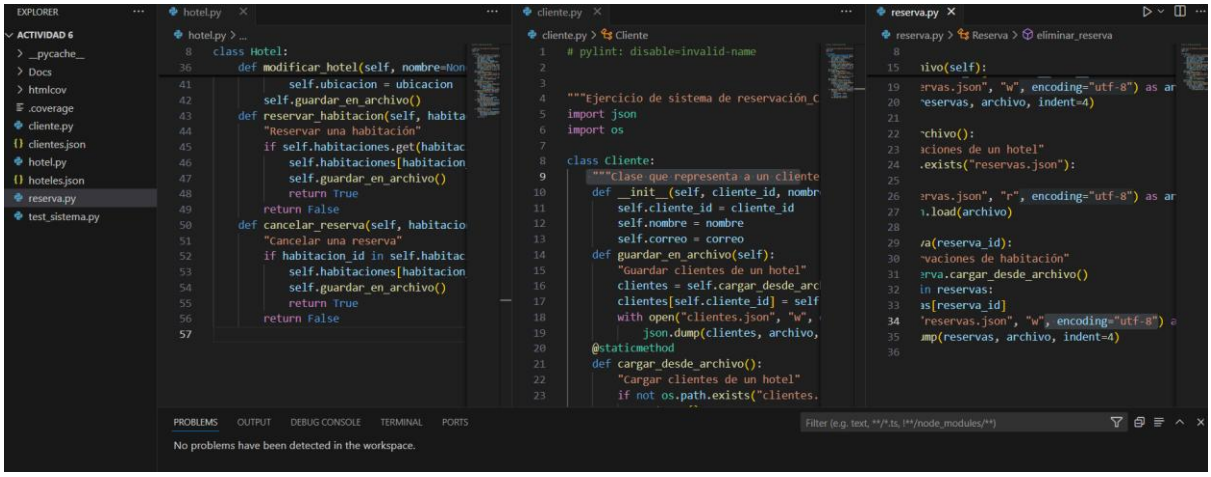

Imagen 2. Resultados sin errores en PyLint de hotel.py, cliente.py, reserva.py.

- Ejemplo si hubiera errores con Flake 8:

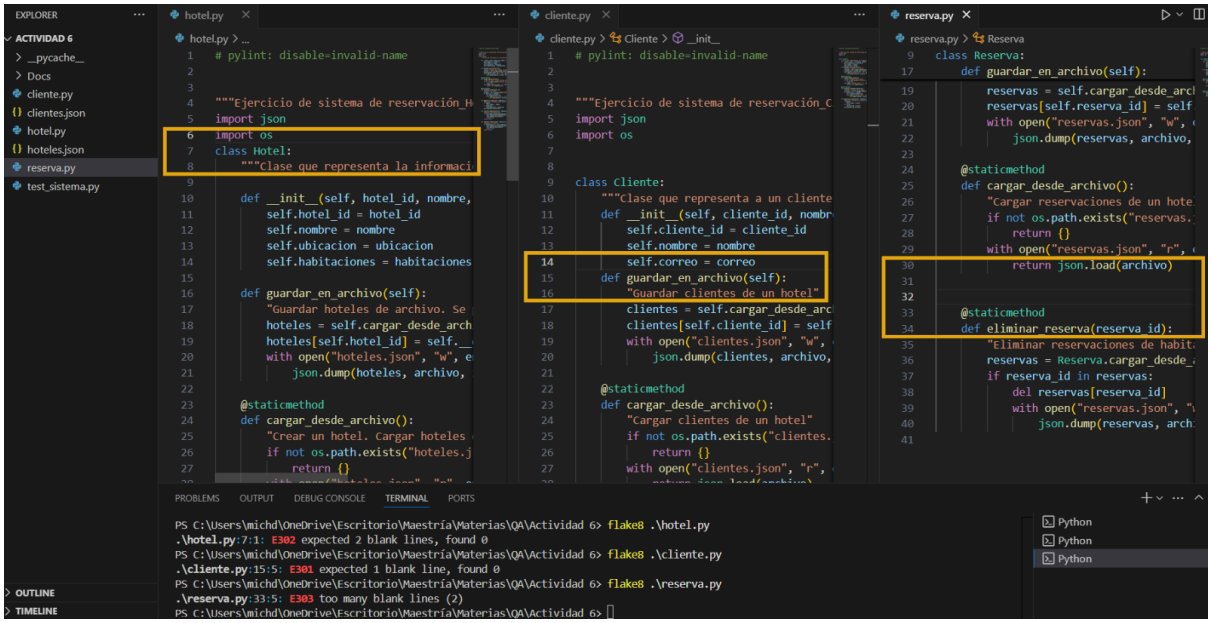

Imagen 3. Resultados con errores en Flake 8 de hotel.py, cliente.py, reserva.py.

- Archivos sin errores de Flake 8:

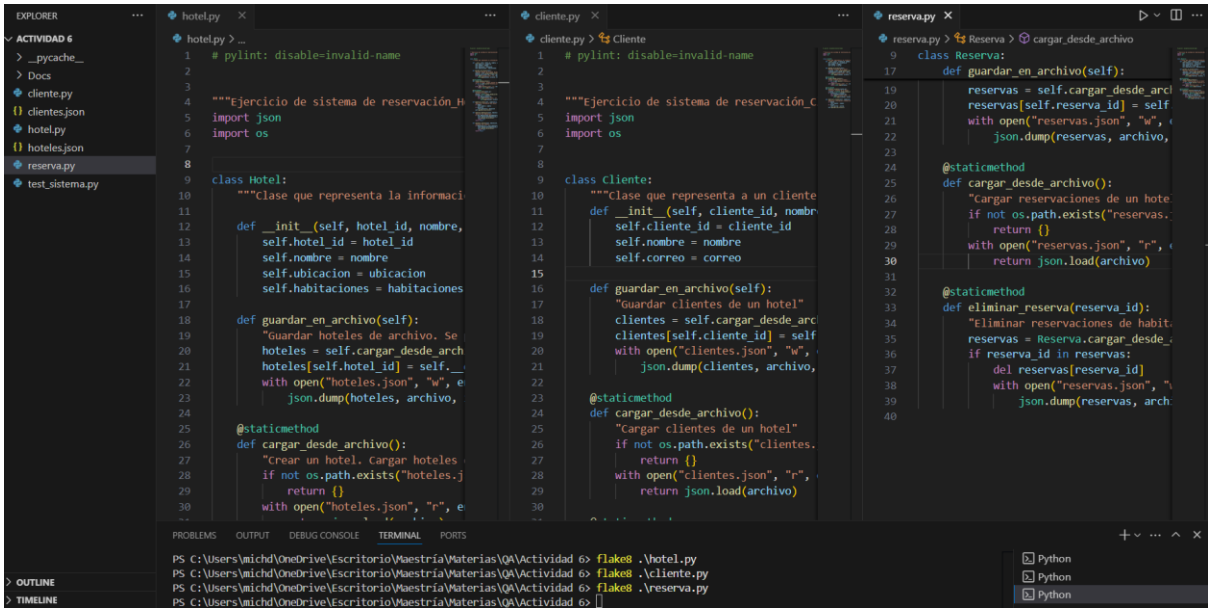

Imagen 4. Resultados sin errores en Flake 8 de hotel.py, cliente.py, reserva.py.

- Se ejecuta el test_sistema.py para las pruebas unitarias a través del módulo unittest. Como se observa en la imagen de abajo, se corren las pruebas sin errores y se generan los .json para el cliente creado (ID del cliente, nombre y correo) así como el .json para la reserva de la habitación asociada al hotel (ID del hotel, nombre del hotel, ubicación y habitaciones disponibles) en donde la reserva también está asociada al cliente generado:

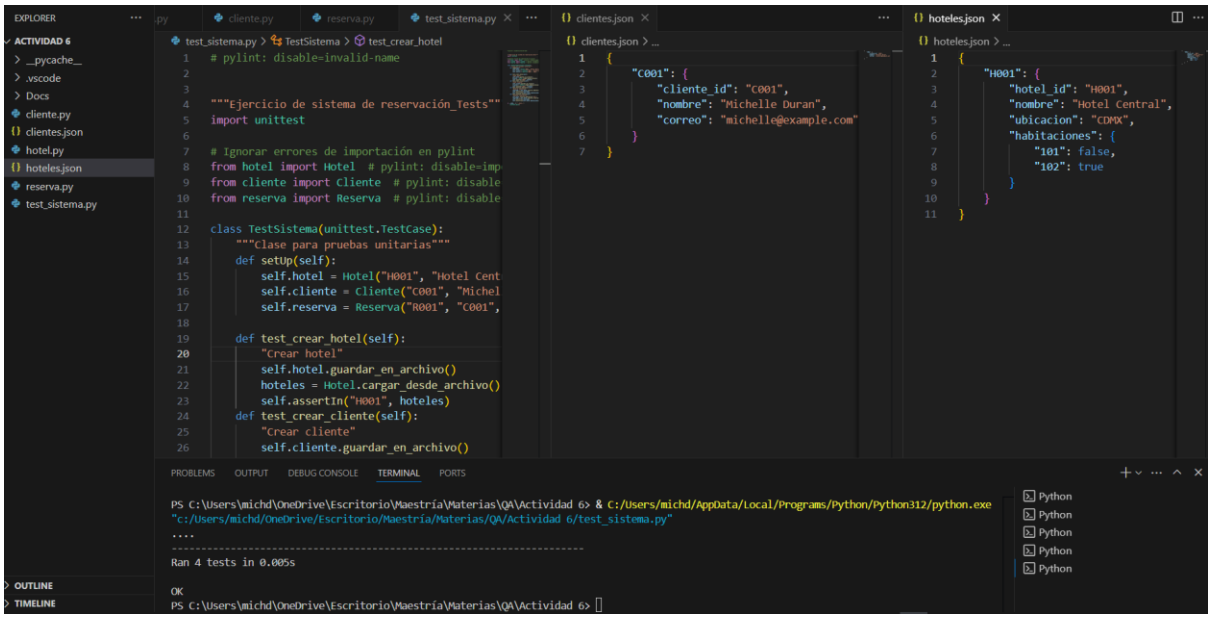

Imagen 5. Resultados de las pruebas unitarias OK.

- Se ejecutan las pruebas de cobertura a través del módulo coverage para saber si la cobertura de código de todas las pruebas unitarias alcanza al menos un 85% de cobertura de líneas (imagen 6) y se obtiene el reporte tanto en la terminal (imagen 7) como en un html (imagen 8) obteniendo un 93% de cobertura:

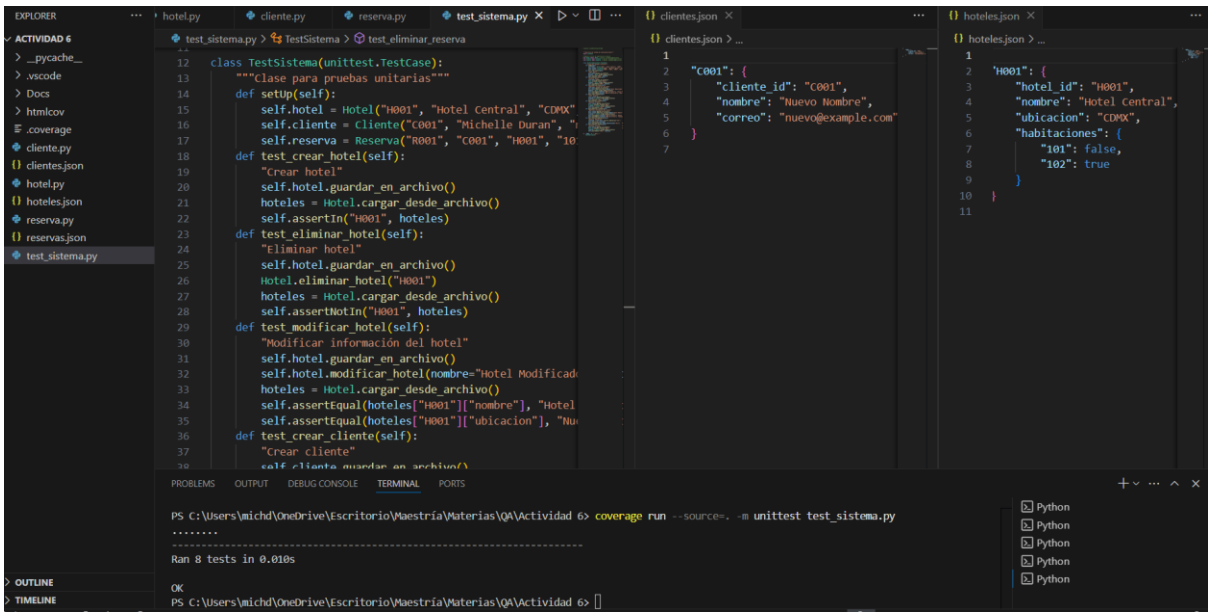

Imagen 6. Ejecución de coverage OK.

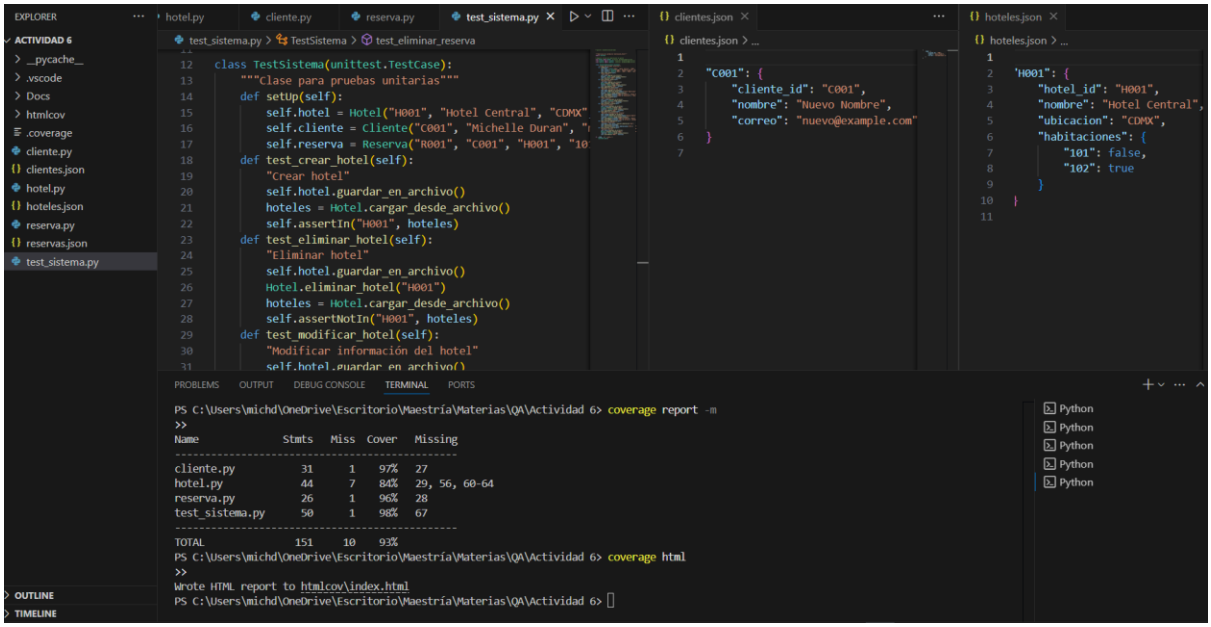

Imagen 7. Cobertura del 93% en terminal.

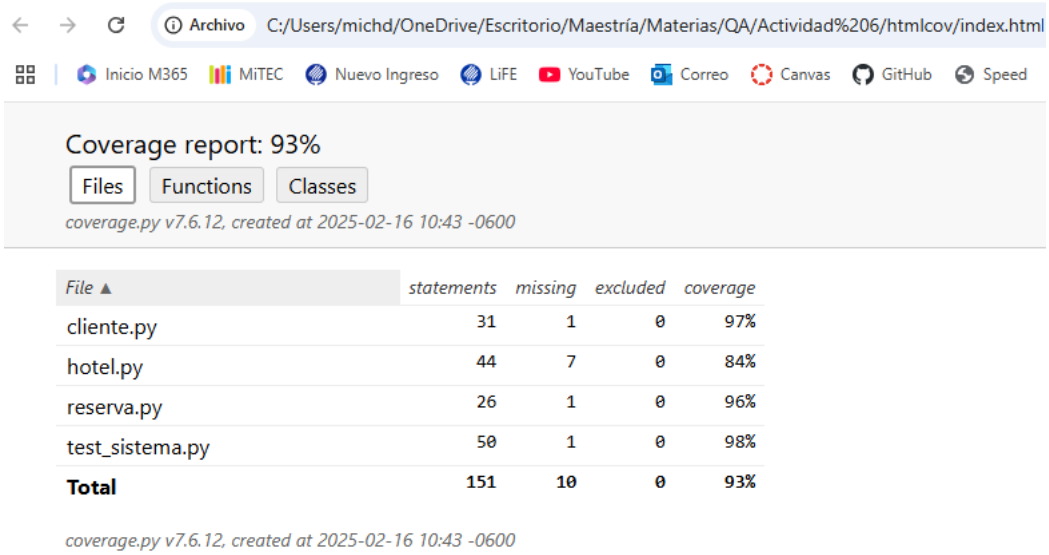

Imagen 8. Cobertura del 93% en index.html.

Liga de GitHub para ver ejercicios:

- https://github.com/MichelleDuranTEC/A01795690_A6.2.git

Referencias Bibliográficas

- Píldoras de programación. (2023, 26 de marzo). *Flake8 en Python, rápido y sencillo 2023.* [Video]. YouTube. https://www.youtube.com/watch?v=G3tSdlBnSgA
- zeroToMasters. (2020, 14 de mayo). *Aprende a generar el CODE COVERAGE paso a paso.* [Video]. YouTube. https://www.youtube.com/watch?v=I3zyO3iPpOE