Instituto Tecnológico y de Estudios Superiores de Monterrey

Maestría en Inteligencia Artificial
Pruebas de software y aseguramiento de la calidad

Alumno: Luis Alfredo Negron Naldos
A01793865

18Febrero 2024

1. **Ruta del GibHub:** https://github.com/PosgradoMNA/actividades-de-aprendizaje-luisnegronnaldos/tree/main/Calidad_software/A01793865_A6.2

2. **Presentación del Código Original**

   Se precedio a desarrollar el código según lo especificado en el trabajo.

```
import json

class Hotel:
   "define la clase Hotel"
    def __init__(self, nombre, direccion, habitaciones_totales):
        self.nombre = nombre
        self.direccion = direccion
        self.habitaciones_totales = habitaciones_totales
        self.habitaciones_reservadas = 0

class Cliente:
    "define la clase Cliente"
    def __init__(self, nombre, apellido, numero_pasaporte):
        self.nombre = nombre
        self.apellido = apellido
        self.numero_pasaporte = numero_pasaporte

class Reserva:
    "define la clase Reserva"
    codigo_reserva = 1

    def __init__(self,codigo_reserva, cliente, hotel, cantidad_personas,
fecha_ingreso, fecha_salida):
        self.codigo_reserva = codigo_reserva
        self.cliente = cliente
        self.hotel = hotel
        self.cantidad_personas = cantidad_personas
        self.fecha_ingreso = fecha_ingreso
        self.fecha_salida = fecha_salida

def cargar_datos(nombre_archivo):
    "define la forma de carga de datos"
    try:
        with open(nombre_archivo, 'r') as file:
            return json.load(file)
    except (FileNotFoundError, json.JSONDecodeError):
        return {}

def guardar_datos(nombre_archivo, datos):
    "define la forma de guardar de datos"
    with open(nombre_archivo, 'w') as file:
        json.dump(datos, file, indent=2)

def mostrar_menu():
    "define el menu principal"
    print("1. Hotels")
```

```
    print("2. Cliente")
    print("3. Reserva")
    print("0. Salir")

def mostrar_submenu_hotels():
    "define el sub menu hoteles"
    print("a. Crear Hotel")
    print("b. Borrar Hotel")
    print("c. Mostrar información del hotel")
    print("d. Modificar información del hotel")
    print("e. Reservar una habitación")
    print("f. Cancelar una reserva")

def mostrar_submenu_cliente():
    "define el sub menu cliente"
    print("a. Crear cliente")
    print("b. Eliminar un cliente")
    print("c. Mostrar información del cliente")
    print("d. Modificar información del cliente")

def mostrar_submenu_reserva():
    "define el sub menu reserva"
    print("a. Crear una Reserva")
    print("b. Cancelar una Reserva")

def main():
    hoteles = cargar_datos("hoteles.json")
    clientes = cargar_datos("clientes.json")
    reservas = cargar_datos("reservas.json")

    while True:
        mostrar_menu()
        opcion_principal = input("Selecciona una opción (0 para salir):
")

        if opcion_principal == "1":
            mostrar_submenu_hotels()
            opcion_hotel = input("Selecciona una opción: ")

            if opcion_hotel == "a":
                # Crear Hotel
                nombre = input("Nombre del hotel: ")
                direccion = input("Dirección del hotel: ")
                habitaciones_totales = int(input("Cantidad de
habitaciones: "))
                hoteles[nombre] = Hotel(nombre, direccion,
habitaciones_totales)

            elif opcion_hotel == "b":
                # Borrar Hotel
                nombre = input("Nombre del hotel a borrar: ")
                if nombre in hoteles:
                    del hoteles[nombre]
                else:
                    print("El hotel no existe.")
```

```
            elif opcion_hotel == "c":
                # Mostrar información del hotel
                nombre = input("Nombre del hotel: ")
                if nombre in hoteles:
                    hotel = hoteles[nombre]
                    print(f"Nombre: {hotel.nombre}")
                    print(f"Dirección: {hotel.direccion}")
                    print(f"Habitaciones Reservadas:
{hotel.habitaciones_reservadas}")
                    print(f"Habitaciones Disponibles:
{hotel.habitaciones_totales - hotel.habitaciones_reservadas}")
                else:
                    print("El hotel no existe.")

            elif opcion_hotel == "d":
                # Modificar información del hotel
                nombre = input("Nombre del hotel a modificar: ")
                if nombre in hoteles:
                    hotel = hoteles[nombre]
                    hotel.nombre = input("Nuevo nombre del hotel: ")
                    hotel.direccion = input("Nueva dirección del hotel:
")
                    hotel.habitaciones_totales = int(input("Nueva
cantidad de habitaciones: "))
                else:
                    print("El hotel no existe.")

            elif opcion_hotel == "e":
                # Reservar una habitación
                nombre_hotel = input("Nombre del hotel: ")
                if nombre_hotel in hoteles:
                    hotel = hoteles[nombre_hotel]
                    # Implementa la lógica para reservar habitación aquí
                else:
                    print("El hotel no existe.")

            elif opcion_hotel == "f":
                # Cancelar una reserva
                # Implementa la lógica para cancelar una reserva aquí
                pass

        elif opcion_principal == "2":
            mostrar_submenu_cliente()
            opcion_cliente = input("Selecciona una opción: ")

            if opcion_cliente == "a":
                # Crear Cliente
                nombre = input("Nombre del cliente: ")
                apellido = input("Apellido del cliente: ")
                numero_pasaporte = input("Número de pasaporte del
cliente: ")
                clientes[numero_pasaporte] = Cliente(nombre, apellido,
numero_pasaporte)
```

```
            elif opcion_cliente == "b":
                # Eliminar un Cliente
                numero_pasaporte = input("Número de pasaporte del cliente
a eliminar: ")
                if numero_pasaporte in clientes:
                    del clientes[numero_pasaporte]
                    print("El cliente eliminado.")
                else:
                    print("El cliente no existe.")

            elif opcion_cliente == "c":
                # Mostrar información del Cliente
                numero_pasaporte = input("Número de pasaporte del
cliente: ")
                if numero_pasaporte in clientes:
                    cliente = clientes[numero_pasaporte]
                    print(f"Nombre: {cliente.nombre}")
                    print(f"Apellido: {cliente.apellido}")
                    print(f"Número de pasaporte:
{cliente.numero_pasaporte}")
                else:
                    print("El cliente no existe.")

            elif opcion_cliente == "d":
                # Modificar información del Cliente
                numero_pasaporte = input("Número de pasaporte del cliente
a modificar: ")
                if numero_pasaporte in clientes:
                    cliente = clientes[numero_pasaporte]
                    cliente.nombre = input("Nuevo nombre del cliente: ")
                    cliente.apellido = input("Nuevo apellido del cliente:
")
                else:
                    print("El cliente no existe.")

        elif opcion_principal == "3":
            mostrar_submenu_reserva()
            opcion_reserva = input("Selecciona una opción: ")

            if opcion_reserva == "a":
                # Crear una Reserva
                codigo_reserva = input("Ingrese el codigo de reserva: ")
                cliente = input("Nombre del cliente: ")
                hotel = input("Apellido del cliente: ")
                cantidad_personas = input("Cantidad de personas: ")
                fecha_ingreso = input("Ingrese la fecha de ingreso
(MM/DD/YY): ")
                fecha_salida = input("Ingrese la fecha de salida
(MM/DD/YY): ")
                reservas[codigo_reserva] =
Reserva(codigo_reserva,cliente,hotel,cantidad_personas,fecha_ingreso,
fecha_salida)
                pass

            elif opcion_reserva == "b":
```

```
                # Cancelar una Reserva
                codigo_reserva = input("Número de reserva a cancelar: ")
                if codigo_reserva in reservas:
                    del reservas[codigo_reserva]
                    print("Reserva Eliminada.")
                else:
                    print("Reserva no existe.")
                pass

        elif opcion_principal == "0":
            # Salir del programa
            guardar_datos("hoteles.json", hoteles)
            guardar_datos("clientes.json", clientes)
            guardar_datos("reservas.json", reservas)
            break

        else:
            print("Opción no válida. Por favor, elige una opción
válida.")

if __name__ == "__main__":
    main()
```

3. Análisis de Errores de Pylint – PEP 8

Se analizo con Pylint y dio los siguiente errores:

(base) MacBook-Air-5:semana6 Luis$ pylint hotel_management.py
********** Module hoteles
hoteles.py:22:0: C0301: Line too long (102/100) (line-too-long)
hoteles.py:107:0: C0301: Line too long (116/100) (line-too-long)
hoteles.py:189:0: C0301: Line too long (126/100) (line-too-long)
hoteles.py:1:0: C0114: Missing module docstring (missing-module-docstring)
hoteles.py:3:0: R0903: Too few public methods (0/2) (too-few-public-methods)
hoteles.py:11:0: R0903: Too few public methods (0/2) (too-few-public-methods)
hoteles.py:22:4: R0913: Too many arguments (7/5) (too-many-arguments)
hoteles.py:18:0: R0903: Too few public methods (0/2) (too-few-public-methods)
hoteles.py:71:0: C0116: Missing function or method docstring (missing-function-docstring)
hoteles.py:71:0: R0914: Too many local variables (19/15) (too-many-locals)
hoteles.py:190:16: W0107: Unnecessary pass statement (unnecessary-pass)
hoteles.py:200:16: W0107: Unnecessary pass statement (unnecessary-pass)
hoteles.py:71:0: R0912: Too many branches (34/12) (too-many-branches)
hoteles.py:71:0: R0915: Too many statements (106/50) (too-many-statements)

Your code has been rated at 9.07/10

Luego revisar el código se logró corregir la mayoria de las obteniendose el siguiente código:

```
"programa para gestion un hotel"
import json
from datetime import datetime

class Hotel:
    "define la clase Hotel"
```

```
    def __init__(self, nombre, direccion, habitaciones_totales):
        self.nombre = nombre
        self.direccion = direccion
        self.habitaciones_totales = habitaciones_totales
        self.habitaciones_reservadas = 0

class Cliente:
    "define la clase Cliente"
    def __init__(self, nombre, apellido, numero_pasaporte):
        self.nombre = nombre
        self.apellido = apellido
        self.numero_pasaporte = numero_pasaporte

class Reserva:
    "define la clase Reserva"
    codigo_reserva = 1

    def __init__(self,codigo_reserva, cliente, hotel, cantidad_personas,
                 fecha_ingreso, fecha_salida):
        self.codigo_reserva = codigo_reserva
        self.cliente = cliente
        self.hotel = hotel
        self.cantidad_personas = cantidad_personas
        self.fecha_ingreso = datetime.strptime(fecha_ingreso, "%m/%d/%Y")
        self.fecha_salida = datetime.strptime(fecha_salida, "%m/%d/%Y")

def cargar_datos(nombre_archivo):
    """define la forma de carga de datos"""
    try:
        with open(nombre_archivo, 'r') as file:
            return json.load(file)
    except (FileNotFoundError, json.JSONDecodeError):
        return {}

def guardar_datos(nombre_archivo, datos):
    """Define como guardar datos"""
    with open(nombre_archivo, 'w') as file:
        json.dump(datos, file, default=serialize, indent=2)

def serialize(obj):
    """serializa los datos de fecha"""
    if isinstance(obj, datetime):
        return obj.strftime('%Y-%m-%d %H:%M:%S')
    return obj.__dict__

def mostrar_menu():
    "define el menu principal"
    print("1. Hotels")
    print("2. Cliente")
    print("3. Reserva")
    print("0. Salir")

def mostrar_submenu_hotels():
    "define el sub menu hoteles"
    print("a. Crear Hotel")
```

```
    print("b. Borrar Hotel")
    print("c. Mostrar información del hotel")
    print("d. Modificar información del hotel")
    print("e. Reservar una habitación")
    print("f. Cancelar una reserva")

def mostrar_submenu_cliente():
    "define el sub menu cliente"
    print("a. Crear cliente")
    print("b. Eliminar un cliente")
    print("c. Mostrar información del cliente")
    print("d. Modificar información del cliente")

def mostrar_submenu_reserva():
    "define el sub menu reserva"
    print("a. Crear una Reserva")
    print("b. Cancelar una Reserva")

def crear_hoteles():
    """ crea hoteles"""
    hoteles = cargar_datos("hoteles.json")
    nombre = input("Nombre del hotel: ")
    direccion = input("Dirección del hotel: ")
    habitaciones_totales = int(input("Cantidad de habitaciones: "))
    hoteles [nombre] = Hotel(nombre, direccion, habitaciones_totales)

def borrar_hoteles():
    """ borrar hoteles"""
    hoteles = cargar_datos("hoteles.json")
    nombre = input("Nombre del hotel a borrar: ")
    if nombre in hoteles:
        del hoteles[nombre]
    else:
        print("El hotel no existe.")

def main():
    """ funcion principal"""
    hoteles = cargar_datos("hoteles.json")
    clientes = cargar_datos("clientes.json")
    reservas = cargar_datos("reservas.json")

    while True:
        mostrar_menu()
        opcion_principal = input("Selecciona una opción (0 para salir):
")

        if opcion_principal == "1":
            mostrar_submenu_hotels()
            opcion_hotel = input("Selecciona una opción: ")

            if opcion_hotel == "a":
                crear_hoteles()

            elif opcion_hotel == "b":
                borrar_hoteles()
```

```
            elif opcion_hotel == "c":
                # Mostrar información del hotel
                nombre = input("Nombre del hotel: ")
                if nombre in hoteles:
                    hotel = hoteles[nombre]
                    print(f"Nombre: {hotel.nombre}")
                    print(f"Dirección: {hotel.direccion}")
                    print(f"Hab. Reservadas:
{hotel.habitaciones_reservadas}")
                    print(f"Hab. Disponible: {hotel.habitaciones_totales
- hotel.habitaciones_reservadas}")
                else:
                    print("El hotel no existe.")

            elif opcion_hotel == "d":
                # Modificar información del hotel
                nombre = input("Nombre del hotel a modificar: ")
                if nombre in hoteles:
                    hotel = hoteles[nombre]
                    hotel.nombre = input("Nuevo nombre del hotel: ")
                    hotel.direccion = input("Nueva dirección del hotel:
")
                    hotel.habitaciones_totales = int(input("Cantidad de
habitaciones: "))
                else:
                    print("El hotel no existe.")

            elif opcion_hotel == "e":
                # Reservar una habitación
                nombre_hotel = input("Nombre del hotel: ")
                if nombre_hotel in hoteles:
                    hotel = hoteles[nombre_hotel]
                    # Implementa la lógica para reservar habitación aquí
                else:
                    print("El hotel no existe.")

            elif opcion_hotel == "f":
                # Cancelar una reserva
                # Implementa la lógica para cancelar una reserva aquí
                pass

        elif opcion_principal == "2":
            mostrar_submenu_cliente()
            opcion_cliente = input("Selecciona una opción: ")

            if opcion_cliente == "a":
                # Crear Cliente
                nombre = input("Nombre del cliente: ")
                apellido = input("Apellido del cliente: ")
                numero_pasaporte = input("Número de pasaporte del
cliente: ")
                clientes[numero_pasaporte] = Cliente(nombre, apellido,
                                                     numero_pasaporte)
```

```
            elif opcion_cliente == "b":
                # Eliminar un Cliente
                numero_pasaporte = input("Pasaporte del cliente a
eliminar: ")
                if numero_pasaporte in clientes:
                    del clientes[numero_pasaporte]
                    print("El cliente eliminado.")
                else:
                    print("El cliente no existe.")

            elif opcion_cliente == "c":
                # Mostrar información del Cliente
                numero_pasaporte = input("Número de pasaporte del
cliente: ")
                if numero_pasaporte in clientes:
                    cliente = clientes[numero_pasaporte]
                    print(f"Nombre: {cliente.nombre}")
                    print(f"Apellido: {cliente.apellido}")
                    print(f"Número de pasaporte:
{cliente.numero_pasaporte}")
                else:
                    print("El cliente no existe.")

            elif opcion_cliente == "d":
                # Modificar información del Cliente
                numero_pasaporte = input("Pasaporte del cliente a
modificar: ")
                if numero_pasaporte in clientes:
                    cliente = clientes[numero_pasaporte]
                    cliente.nombre = input("Nuevo nombre del cliente: ")
                    cliente.apellido = input("Nuevo apellido del cliente:
")
                else:
                    print("El cliente no existe.")

        elif opcion_principal == "3":
            mostrar_submenu_reserva()
            opcion_reserva = input("Selecciona una opción: ")

            if opcion_reserva == "a":
                # Crear una Reserva
                codigo_reserva = input("Ingrese el codigo de reserva: ")
                cliente = input("Nombre del cliente: ")
                hotel = input("Apellido del cliente: ")
                cantidad_personas = input("Cantidad de personas: ")
                fecha_ingreso = input("Fecha de ingreso (MM/DD/YYYY): ")
                fecha_salida = input("Fecha de salida (MM/DD/YYYY): ")
                reservas[codigo_reserva] =
Reserva(codigo_reserva,cliente,

hotel,cantidad_personas,
                                                    fecha_ingreso,
fecha_salida)
                #pass
```

```
            elif opcion_reserva == "b":
                # Cancelar una Reserva
                codigo_reserva = input("Número de reserva a cancelar: ")
                if codigo_reserva in reservas:
                    del reservas[codigo_reserva]
                    print("Reserva Eliminada.")
                else:
                    print("Reserva no existe.")
                #pass

        elif opcion_principal == "0":
            # Salir del programa
            guardar_datos("hoteles.json", hoteles)
            guardar_datos("clientes.json", clientes)
            guardar_datos("reservas.json", reservas)
            break

        else:
            print("Opción no válida. Por favor, elige una opción
válida.")

if __name__ == "__main__":
    main()
```

Con las siguientes observaciones:

(base) MacBook-Air-5:semana6 Luis$ pylint hotel_management.py
********** Module hotels
hoteles.py:97:0: R0912: Too many branches (32/12) (too-many-branches)
hoteles.py:97:0: R0915: Too many statements (97/50) (too-many-statements)

Your code has been rated at 9.50/10 (previous run: 9.83/10, +0.00)

4. Análisis con Flake 8

MacBook-Air-5:semana6 Luis$ flake8 hotel_management.py --statistics
hoteles.py:5:1: E302 expected 2 blank lines, found 1
hoteles.py:13:1: E302 expected 2 blank lines, found 1
hoteles.py:20:1: E302 expected 2 blank lines, found 1
hoteles.py:24:22: E231 missing whitespace after ','
hoteles.py:33:1: E302 expected 2 blank lines, found 1
hoteles.py:41:1: E302 expected 2 blank lines, found 1
hoteles.py:46:1: E302 expected 2 blank lines, found 1
hoteles.py:52:1: E302 expected 2 blank lines, found 1
hoteles.py:59:1: E302 expected 2 blank lines, found 1
hoteles.py:68:1: E302 expected 2 blank lines, found 1
hoteles.py:75:1: E302 expected 2 blank lines, found 1
hoteles.py:80:1: E302 expected 2 blank lines, found 1
hoteles.py:86:12: E211 whitespace before '['
hoteles.py:88:1: E302 expected 2 blank lines, found 1
hoteles.py:97:1: E302 expected 2 blank lines, found 1

```
hoteles.py:125:80: E501 line too long (107 > 79 characters)
hoteles.py:136:80: E501 line too long (89 > 79 characters)
hoteles.py:208:66: E231 missing whitespace after ','
hoteles.py:209:57: E231 missing whitespace after ','
hoteles.py:211:17: E265 block comment should start with '# '
hoteles.py:221:17: E265 block comment should start with '# '
hoteles.py:233:1: E305 expected 2 blank lines after class or function definition, found 1
1     E211 whitespace before '['
3     E231 missing whitespace after ','
2     E265 block comment should start with '# '
13    E302 expected 2 blank lines, found 1
1     E305 expected 2 blank lines after class or function definition, found 1
2     E501 line too long (107 > 79 characters)
```

Se procedió a corregir el código con las observaciones y se obtuvo el siguiente resultado:

```
(base) MacBook-Air-5:semana6 Luis$ flake8 hotel_management.py –statistics
hoteles.py:138:80: E501 line too long (107 > 79 characters)
hoteles.py:149:80: E501 line too long (89 > 79 characters)
2     E501 line too long (107 > 79 characters)
```

5. Creación de Unit Test

Se crearon 15 pruebas el siguiente archive de pruebas: test_hotel_management.py, de las cuales se crearon 5 casos negativos:

- test_cargar_datos_archivo_invalido
- test_creacion_hotel_invalida
- test_creacion_reserva_fecha_invalida
- test_eliminar_cliente_no_existente
- test_mostrar_informacion_hotel_no_existente

con el siguiente código:

```python
import unittest
from datetime import datetime
from hotel_management import Hotel, Cliente, Reserva, cargar_datos,
guardar_datos

class TestHotelManagement(unittest.TestCase):

    def test_creacion_hotel(self):
        hotel = Hotel("Hotel Ejemplo", "Direccion Ejemplo", 10)
        self.assertEqual(hotel.nombre, "Hotel Ejemplo")
        self.assertEqual(hotel.direccion, "Direccion Ejemplo")
        self.assertEqual(hotel.habitaciones_totales, 10)
        self.assertEqual(hotel.habitaciones_reservadas, 0)

    def test_creacion_cliente(self):
        cliente = Cliente("Juan", "Perez", "123456789")
        self.assertEqual(cliente.nombre, "Juan")
        self.assertEqual(cliente.apellido, "Perez")
        self.assertEqual(cliente.numero_pasaporte, "123456789")

    def test_creacion_reserva(self):
```

```
        fecha_ingreso = datetime.strptime("01/01/2023", "%m/%d/%Y")
        fecha_salida = datetime.strptime("01/05/2023", "%m/%d/%Y")
        hotel = Hotel("Hotel Ejemplo", "Direccion Ejemplo", 10)
        cliente = Cliente("Juan", "Perez", "123456789")
        reserva = Reserva(1, cliente, hotel, 2, fecha_ingreso,
fecha_salida)
        self.assertEqual(reserva.codigo_reserva, 1)
        self.assertEqual(reserva.cliente, cliente)
        self.assertEqual(reserva.hotel, hotel)
        self.assertEqual(reserva.cantidad_personas, 2)
        self.assertEqual(reserva.fecha_ingreso, fecha_ingreso)
        self.assertEqual(reserva.fecha_salida, fecha_salida)

    def test_cargar_guardar_datos(self):
        datos_originales = {"nombre": "Hotel Ejemplo", "direccion":
"Direccion Ejemplo", "habitaciones_totales": 10}
        guardar_datos("prueba.json", datos_originales)
        datos_cargados = cargar_datos("prueba.json")
        self.assertEqual(datos_cargados, datos_originales)

        # Limpia el archivo de prueba después de la prueba
        guardar_datos("prueba.json", {})

    def test_modificar_hotel(self):
        hotel = Hotel("Hotel Ejemplo", "Direccion Ejemplo", 10)
        hotel.nombre = "Nuevo Hotel"
        hotel.direccion = "Nueva Direccion"
        hotel.habitaciones_totales = 15
        self.assertEqual(hotel.nombre, "Nuevo Hotel")
        self.assertEqual(hotel.direccion, "Nueva Direccion")
        self.assertEqual(hotel.habitaciones_totales, 15)

    def test_eliminar_cliente(self):
        cliente = Cliente("Juan", "Perez", "123456789")
        clientes = {"123456789": cliente}
        del clientes["123456789"]
        self.assertNotIn("123456789", clientes)

    def test_reservar_habitacion(self):
        hotel = Hotel("Hotel Ejemplo", "Direccion Ejemplo", 10)
        hotel.habitaciones_reservadas = 5
        self.assertEqual(hotel.habitaciones_reservadas, 5)
        # Realiza la lógica de reserva aquí y verifica que las
habitaciones reservadas se actualicen correctamente

    def test_cancelar_reserva(self):
        reserva = Reserva(1, Cliente("Juan", "Perez", "123456789"),
Hotel("Hotel Ejemplo", "Direccion Ejemplo", 10), 2, datetime.now(),
datetime.now())
        reservas = {"1": reserva}
        del reservas["1"]
        self.assertNotIn("1", reservas)

    def test_mostrar_informacion_cliente_no_existente(self):
        clientes = {}
```

```
        self.assertFalse("123456789" in clientes)  # Asegura que el
cliente no exista

    def test_mostrar_informacion_hotel_no_existente(self):
        hoteles = {}
        self.assertFalse("Hotel Inexistente" in hoteles)  # Asegura que
el hotel no exista

    def test_guardar_datos_error(self):
        datos_originales = {"nombre": "Hotel Ejemplo", "direccion":
"Direccion Ejemplo", "habitaciones_totales": 10}
        with self.assertRaises(Exception):  # Intencionalmente causando
un error al guardar datos
            guardar_datos("/ruta/inexistente/prueba.json",
datos_originales)

    def test_cargar_datos_archivo_invalido(self):
        datos_cargados = cargar_datos("archivo_invalido.json")
        self.assertEqual(datos_cargados, {})  # Asegura que se devuelva
un diccionario vacío para un archivo no válido

    def test_creacion_hotel_invalida(self):
        with self.assertRaises(ValueError):  # Intencionalmente causando
un error al crear hotel con habitaciones totales negativas
            hotel = Hotel("Hotel Ejemplo", "Direccion Ejemplo", -5)

    def test_creacion_reserva_fecha_invalida(self):
        fecha_ingreso = datetime.strptime("01/01/2023", "%m/%d/%Y")
        fecha_salida = datetime.strptime("01/01/2023", "%m/%d/%Y")
        hotel = Hotel("Hotel Ejemplo", "Direccion Ejemplo", 10)
        cliente = Cliente("Juan", "Perez", "123456789")
        with self.assertRaises(ValueError):  # Intencionalmente causando
un error al crear reserva con fechas iguales
            reserva = Reserva(1, cliente, hotel, 2, fecha_ingreso,
fecha_salida)

    def test_eliminar_cliente_no_existente(self):
        clientes = {}
        with self.assertRaises(KeyError):  # Intencionalmente causando un
error al eliminar cliente no existente
            del clientes["123456789"]

    def test_mostrar_informacion_hotel_no_existente(self):
        hoteles = {}
        with self.assertRaises(KeyError):  # Intencionalmente causando un
error al intentar acceder a información de hotel no existente
            hotel = hoteles["Hotel Inexistente"]

    def test_cargar_datos_archivo_invalido(self):
        with self.assertRaises((FileNotFoundError,
json.JSONDecodeError)):  # Intencionalmente causando un error al cargar
datos desde un archivo inexistente o inválido
            datos_cargados = cargar_datos("archivo_invalido.json")

if __name__ == '__main__':
```

```
    unittest.main()
```

Obteniendose el siguiente resultado:

(base) MacBook-Air-5:semana6 Luis$ python -m unittest test_hotel_management.py
.....
--
Ran 15 tests in 0.003s

OK

6. **Cobertura**

Se procedio a generar el reporte de cobertura como se muestra a continuación.

(base) MacBook-Air-5:semana6 Luis$ coverage report -m

Name	Stmts	Miss	Cover	Missing
hotel_management.py	153	98	36%	35, 43-44, 55-57, 62-65, 70-75, 80-83, 88-89, 94-231, 235
test_hotel_management.py	91	15	84%	26-31, 65-67, 74-75, 83-84, 110, 113
TOTAL	244	113	53%	

(base) MacBook-Air-5:semana6 Luis$