Instituto Tecnológico y de Estudios

Superiores de Monterrey

Maestría en Inteligencia Artificial Aplicada (MNA)

Pruebas de software y aseguramiento de la calidad (Grupo 10)

Tarea Individual - Actividad 6.3

Ejercicio de programación 3

Estudiante:

Ana Cristina Torres Cordero

A00831285

12 de febrero del 2025

Monterrey. Nuevo León

Ejercicio de programación

1. Reservation System

Se creó el archivo reservation_system siguiendo las instrucciones de la actividad. El archivo implementa un sistema de reservas de hotel con clases para gestionar hoteles, clientes y reservas. Incluye funcionalidad para crear, modificar y eliminar estas entidades, así como datos persistentes en archivos JSON.

Al correr el código, se corrieron las 13 pruebas exitosamente y se muestran los mensajes de error correspondientes a las pruebas:

```
!coverage run -m unittest reservation_system.py

Error: Customer ID already exists.
Error: Hotel ID already exists.
Hotel not found.
Customer not found.
.............
----------------------------------------------------------------------
Ran 13 tests in 0.025s

OK
```

Estos mensajes son de utilidad a la hora de correr el programa, lo que nos muestra que las pruebas funcionan correctamente.

Además, se corrieron las pruebas con PyLint y Flake8 y se corrigieron los errores correspondientes para obtener resultados satisfactorios:

```
!flake8 reservation_system.py

!pylint reservation_system.py


--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
```

Al tener el código listo y en línea con el estándar PEP8, se creó un nuevo código en Python llamado test_scripts.py que genera archivos JSON con datos iniciales para el sistema. Se cargaron datos de hoteles, clientes y reservaciones:

```
%run test_scripts.py

Archivo hotels.json ya existe.
Archivo customers.json ya existe.
Archivo reservations.json ya existe.
```

Por último, se creó un nuevo script para validar el sistema. Este script muestra los datos iniciales y luego crea un nuevo hotel, cliente y reservación. En cada paso, verifica que los resultados se muestren correctamente. También se actualiza una reservación y se cancela:

```
%run reservation_system_validation.py

Hoteles:
[{'hotel_id': 1, 'name': 'Hotel Paradise', 'location': 'New York', 'rooms': 100}]

Clientes:
[{'customer_id': 1, 'name': 'John Doe', 'email': 'john@example.com'}]

Reservaciones:
[{'reservation_id': 1, 'customer_id': 1, 'hotel_id': 1, 'room_number': 50}]

Creando un nuevo hotel...

Hoteles actualizados:
[{'hotel_id': 1, 'name': 'Hotel Paradise', 'location': 'New York', 'rooms': 100}, {'hotel_id': 3, 'name': 'Mountai
n Lodge', 'location': 'Denver', 'rooms': 50}]


Creando un nuevo cliente...

Clientes actualizados:
[{'customer_id': 1, 'name': 'John Doe', 'email': 'john@example.com'}, {'customer_id': 3, 'name': 'Alice Johnson',
'email': 'alice@example.com'}]

Creando una nueva reservación...

Reservaciones actualizadas:
[{'reservation_id': 1, 'customer_id': 1, 'hotel_id': 1, 'room_number': 50}, {'reservation_id': 3, 'customer_id':
3, 'hotel_id': 3, 'room_number': 25}]

Cancelando la reservación con ID 1...

Reservaciones después de la cancelación:
[{'reservation_id': 3, 'customer_id': 3, 'hotel_id': 3, 'room_number': 25}]
```

El programa final cumple con todas las especificaciones y requisitos plasmados en la actividad y se adjuntan en el repositorio de GitHub. El coverage report nos muestra que se cubre al menos el 85% de las líneas:

```
!coverage report

Name                     Stmts   Miss  Cover
------------------------------------------
reservation_system.py      195     30    85%
------------------------------------------
TOTAL                      195     30    85%
```