



6.2 Ejercicio de programación 3 y pruebas de unidad

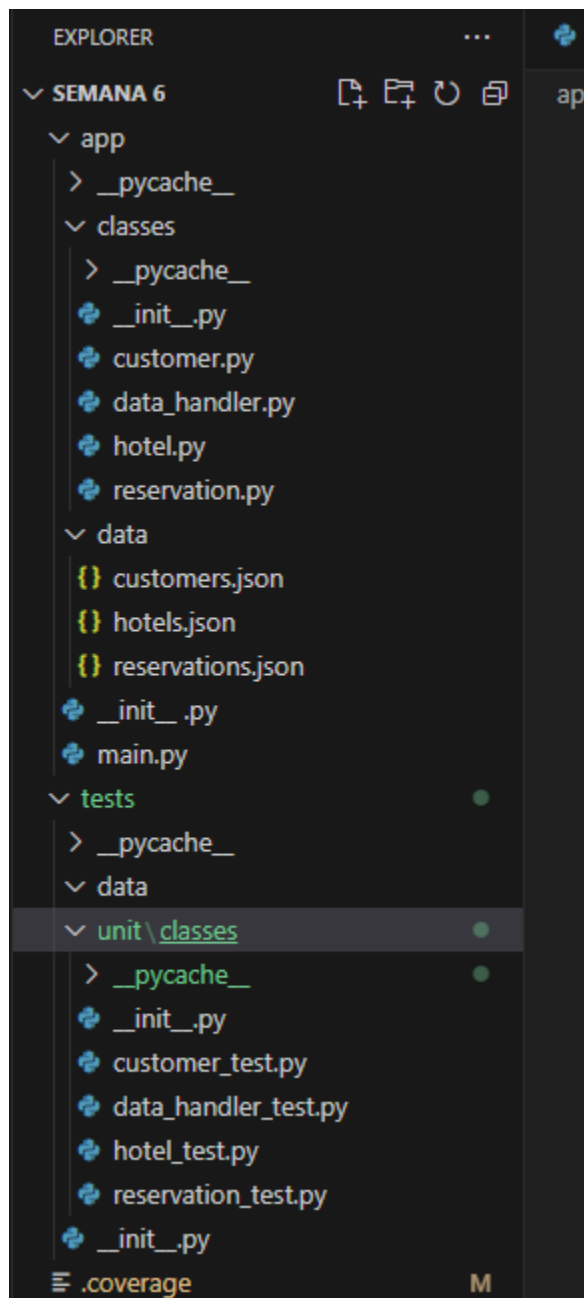
Pruebas de software y aseguramiento de la calidad

Andre M. Hernández Bornn

A01795190

12 febrero 2025

Estructura de aplicación



Clase Hotel

```
hotel.py X main.py M customer.py reservation.py data_handler.py ho
app > classes > hotel.py > Hotel > make_reservation

1  '''Archivo para crear clase de Hotel'''
2
3
4  import os
5  from app.classes.reservation import Reservation
6  from app.classes.data_handler import DataHandler
7
8
9  class Hotel():
10     '''Clase para manejar hoteles'''
11
12     def __init__(self, data_handler):
13         base_path = os.path.dirname(os.path.abspath(__file__))
14         file_path = os.path.join(base_path, "..", "data", "reservations.json")
15         self.data_handler = data_handler
16         reservation_data_handler = DataHandler(file_path)
17         self.reservation_instance = Reservation(reservation_data_handler)
18
19     def create_hotel(self, name, country, state, address):
20         '''Metodo para crear hotel'''
21         try:
22             hotel = {
23                 "name": name,
24                 "country": country,
25                 "state": state,
26                 "address": address,
27             }
28             if any(
29                 self.data_handler.is_missing(hotel[key])
30                 for key in ["name", "country", "state", "address"]
31             ):
32                 raise ValueError("Missing data")
33             new_hotel_list = self.data_handler.create_registry(hotel)
34             return new_hotel_list
35         except ValueError:
36             return "Missing data"
37
38     def delete_hotel(self, hotel_id):
39         '''Metodo para borrar hotel'''
40         try:
41             if self.data_handler.is_missing(hotel_id):
42                 raise ValueError("Missing data")
43             all_hotels = self.data_handler.delete_from_file(hotel_id)
44             return all_hotels
45         except ValueError:
46             return "Missing data"
47
48     def get_hotel_info(self, hotel_id):
49         '''Metodo para obtener los datos de un hotel'''
50         try:
51             if self.data_handler.is_missing(hotel_id):
52                 raise ValueError("Missing data")
53             hotel_info = self.data_handler.get_data_by_id(hotel_id)
54             return hotel_info
55         except ValueError:
56             return "Missing data"
57
```

```
hotel.py × main.py M customer.py reservation.py data_handler.py hotel_test.py
app > classes > hotel.py > Hotel > modify_hotel
9 class Hotel():
38 def delete_hotel(self, hotel_id):
46     return "Missing data"
47
48 def get_hotel_info(self, hotel_id):
49     '''Metodo para obtener los datos de un hotel'''
50     try:
51         if self.data_handler.is_missing(hotel_id):
52             raise ValueError("Missing data")
53         hotel_info = self.data_handler.get_data_by_id(hotel_id)
54         return hotel_info
55     except ValueError:
56         return "Missing data"
57
58 def modify_hotel(self, hotel):
59     '''Metodo para modificar un hotel'''
60     try:
61         if any(
62             self.data_handler.is_missing(hotel[key])
63             for key in ["id", "name", "country", "state", "address"]
64         ):
65             raise ValueError("Missing data")
66         all_hotels = self.data_handler.modify_file(hotel)
67         return all_hotels
68     except ValueError:
69         return "Missing data"
70
71 def make_reservation(
72     self, customer_id, hotel_id, check_in_date, check_out_date
73 ):
74     '''Metodo para crear
75     una reservación'''
76     try:
77         reservation = {
78             "hotel_id": hotel_id,
79             "customer_id": customer_id,
80             "check_in_date": check_in_date,
81             "check_out_date": check_out_date
82         }
83         reservation_list = self.reservation_instance.create_reservation(
84             reservation, self
85         )
86         if reservation_list == (
87             "Hotel or Customer with specified ID does not exist"
88         ):
89             raise ValueError
90         return reservation_list
91     except ValueError:
92         return "Failed creating reservation"
93
94 def cancel_reservation(self, reservation_id):
95     '''Metodo para cancelar una reservación'''
96     reservation_list = self.reservation_instance.cancel_reservation(
97         reservation_id
98     )
99     return reservation_list
100
```

Clase Customer

```
hotel.py  main.py M  customer.py X  reservation.py  data_handler.py  hotel_te

app > classes > customer.py > Customer > get_customer_data

1  '''Archivo para crear clase de clientes'''
2
3
4  class Customer():
5      '''Clase para manejar clientes'''
6
7      def __init__(self, data_handler):
8          self.data_handler = data_handler
9
10     def create_customer(self, name, last_name):
11         '''Metodo para crear clientes'''
12         try:
13             customer = {
14                 "name": name,
15                 "lastName": last_name,
16             }
17             if any(
18                 self.data_handler.is_missing(customer[key])
19                 for key in ["name", "lastName"]
20             ):
21                 raise ValueError("Missing data")
22             new_customer_list = self.data_handler.create_registry(customer)
23             return new_customer_list
24         except ValueError:
25             return "Missing data"
26
27     def delete_customer(self, customer_id):
28         '''Metodo para borrar un cliente'''
29         try:
30             if (
31                 self.data_handler.is_missing(customer_id)
32             ):
33                 raise ValueError("Missing data")
34             all_customers = self.data_handler.delete_from_file(customer_id)
35             return all_customers
36         except ValueError:
37             return "Missing data"
38
39     def modify_customer(self, customer):
40         '''Metodo para modificar un cliente'''
41         try:
42             if any(
43                 self.data_handler.is_missing(customer[key])
44                 for key in ["id", "name", "lastName"]
45             ):
46                 raise ValueError("Missing data")
47             all_customers = self.data_handler.modify_file(customer)
48             return all_customers
49         except ValueError:
50             return "Missing data"
51
52     def get_customer_data(self, customer_id):
53         '''Metodo para obtener datos de un cliente'''
54         try:
55             if self.data_handler.is_missing(customer_id):
56                 raise ValueError("Missing data")
57             customer_info = self.data_handler.get_data_by_id(customer_id)
58             return customer_info
59         except ValueError:
60             return "Missing data"
61
```

Clase reservation

```
hotel.py  main.py M  customer.py  reservation.py X  data_handler.py  ho

app > classes > reservation.py > Reservation > get_reservation_by_id

1  '''Archivo para crear clase de Reservation'''
2
3  import os
4
5  from app.classes.customer import Customer
6  from app.classes.data_handler import DataHandler
7
8
9  class Reservation():
10     '''Clase para manejar reservaciones'''
11
12     def __init__(self, data_handler):
13         base_dir = os.path.dirname(
14             os.path.abspath(__file__)
15         )
16         customer_file_path = os.path.join(
17             base_dir, "..", "data", "customers.json"
18         )
19         self.data_handler = data_handler
20         customer_data_handler = DataHandler(customer_file_path)
21         self.customer_instance = Customer(customer_data_handler)
22
23     def create_reservation(
24         self, reservation, hotel_instance
25     ):
26         '''Metodo para crear reservación'''
27         try:
28             if any(
29                 self.data_handler.is_missing(reservation[key])
30                 for key in [
31                     "hotel_id",
32                     "customer_id",
33                     "check_in_date",
34                     "check_out_date"
35                 ]
36             ):
37                 raise ValueError("Missing data")
38             customer = self.customer_instance.get_customer_data(
39                 reservation["customer_id"]
40             )
41             hotel = hotel_instance.get_hotel_info(
42                 reservation["customer_id"]
43             )
44             if (
45                 customer == "No data found"
46                 or hotel == "No data found"
47             ):
48                 return "Hotel or Customer with specified ID does not exist"
49             reservation = {
50                 "hotelID": reservation["hotel_id"],
51                 "CustomerID": reservation["customer_id"],
52                 "checkInDate": reservation["check_in_date"],
53                 "checkOutDate": reservation["check_out_date"],
54                 "Status": "confirmed"
55             }
56             new_reservation_list = self.data_handler.create_registry(
57                 reservation
58             )
59             return new_reservation_list
60         except ValueError:
61             return "Missing data"
```

hotel.py

main.py M

customer.py

reservation.py X

data_handler.py

app > classes > reservation.py > Reservation > get_reservation_by_id

```
9 class Reservation():
23     def create_reservation(
43         ):
44         if (
45             customer == "No data found"
46             or hotel == "No data found"
47         ):
48             return "Hotel or Customer with specified ID does not exist"
49         reservation = {
50             "hotelID": reservation["hotel_id"],
51             "CustomerID": reservation["customer_id"],
52             "checkInDate": reservation["check_in_date"],
53             "checkOutDate": reservation["check_out_date"],
54             "Status": "confirmed"
55         }
56         new_reservation_list = self.data_handler.create_registry(
57             reservation
58         )
59         return new_reservation_list
60     except ValueError:
61         return "Missing data"
62
63     def cancel_reservation(self, reservation_id):
64         '''Metodo para cancelar reservación'''
65         reservation_to_modify = self.get_reservation_by_id(reservation_id)
66         if reservation_to_modify == "No data found":
67             return "Id does not exist"
68         reservation_to_modify["Status"] = "canceled"
69         modified_reservation_list = self.data_handler.modify_file(
70             reservation_to_modify
71         )
72         return modified_reservation_list
73
74     def get_reservation_by_id(self, reservation_id):
75         '''Metodo para obtener info de reservación'''
76         try:
77             if self.data_handler.is_missing(reservation_id):
78                 raise ValueError("Missing data")
79             reservation_info = self.data_handler.get_data_by_id(reservation_id)
80             return reservation_info
81         except ValueError:
82             return "Missing data"
83
```

Clase datahandler

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py X  hotel_test.py

app > classes > data_handler.py > DataHandler > create_registry

1  '''Archivo para crear clase de DataHandler'''
2
3  import json
4
5
6  class DataHandler():
7      '''Clase para manejar datos en los archivos'''
8
9      def __init__(self, file):
10         self.file = file
11
12     def create_registry(self, data):
13         '''Metodo para crear un nuevo registro en el archivo'''
14         file_json = self.file_as_json()
15         my_id = self.create_id(file_json)
16         data["id"] = my_id
17         file_json.append(data)
18         file_json = self.write_to_file(file_json)
19         return file_json
20
21     def modify_file(self, data):
22         '''Metodo para modificar un registro en el archivo'''
23         my_id = data["id"]
24         file_json = self.file_as_json()
25         index = self.find_index_of_array(file_json, my_id)
26         if index < 0:
27             return file_json
28         file_json[index] = data
29         file_json = self.write_to_file(file_json)
30         return file_json
31
32     def get_data_by_id(self, my_id):
33         '''Metodo para obtener un registro del archivo'''
34         file_json = self.file_as_json()
35         index = self.find_index_of_array(file_json, my_id)
36         if index < 0:
37             return "No data found"
38         data = file_json[index]
39         return data
40
41     def delete_from_file(self, my_id):
42         '''Metodo para borrar un registro del archivo'''
43         file_json = self.file_as_json()
44         json_file = self.delete_from_json(my_id, file_json)
45         self.write_to_file(json_file)
46         return json_file
47
48     def file_as_json(self):
49         '''Metodo para leer al archivo y convertirlo a json'''
50         try:
51             with open(self.file, encoding="utf-8") as json_file:
52                 json_data = json.load(json_file)
53                 return json_data
54         except FileNotFoundError:
55             print("File does not exists")
56             return "File Does not exists"
57         except json.decoder.JSONDecodeError as e:
58             print("Empty Json, give proper format to file")
59             raise e
60
61     def filter_json_file(self, my_id, json_file):
62         '''Metodo para filtrar un json
63         y obtener los registros con un id especifico'''
64         filtered_data = [x for x in json_file if x['id'] == my_id]
65         return filtered_data
```



```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py X  hotel_test.py

app > classes > data_handler.py > DataHandler > create_registry
6  class DataHandler():
41  def delete_from_file(self, my_id):
46      return json_file
47
48  def file_as_json(self):
49      '''Metodo para leer al archivo y convertirlo a json'''
50      try:
51          with open(self.file, encoding="utf-8") as json_file:
52              json_data = json.load(json_file)
53              return json_data
54      except FileNotFoundError:
55          print("File does not exists")
56          return "File Does not exists"
57      except json.decoder.JSONDecodeError as e:
58          print("Empty Json, give propper format to file")
59          raise e
60
61  def filter_json_file(self, my_id, json_file):
62      '''Metodo para filtrar un json
63      y obtener los registros con un id especifico'''
64      filtered_data = [x for x in json_file if x['id'] == my_id]
65      return filtered_data
66
67  def delete_from_json(self, my_id, json_file):
68      '''Metodo para borrar un elemento especifico del json'''
69      index = self.find_index_of_array(json_file, my_id)
70      del json_file[index]
71      return json_file
72
73  def find_index_of_array(self, json_file, my_id):
74      '''Metodo para obtener el indice del
75      registro con el id proporcionado'''
76      for i, dic in enumerate(json_file):
77          if dic["id"] == my_id:
78              return i
79      return -1
80
81  def write_to_file(self, original_json_file):
82      '''Metodo para escribir en el archivo'''
83      try:
84          with open(self.file, "w", encoding="utf-8") as json_file:
85              json.dump(original_json_file, json_file)
86          with open(self.file, encoding="utf-8") as json_file:
87              return json.load(json_file)
88      except TypeError:
89          print("Error writing to file")
90          return "Error writing to file"
91
92  def create_id(self, json_file):
93      '''Metodo para crear un ID'''
94      my_id = 1
95      if len(json_file) < 1:
96          return my_id
97      my_id = json_file[len(json_file) - 1]["id"] + 1
98      return my_id
99
100  def is_missing(self, value):
101      '''Metodo para validar si un dato viene vacio'''
102      return value is None or value == ""
103
```

Código de control

```
Terminal  Help
hotel.py  main.py M X  customer.py  reservation.py  data_handler.py
app > main.py > ...

1  '''Functionality Test'''
2  import os
3
4  from app.classes.hotel import Hotel
5  from app.classes.data_handler import DataHandler
6  from app.classes.customer import Customer
7
8
9  BASE_DIR = os.path.dirname(os.path.abspath(__file__))
10 HOTEL_FILE_PATH = os.path.join(BASE_DIR, "data", "hotels.json")
11 CUSTOMER_FILE_PATH = os.path.join(BASE_DIR, "data", "customers.json")
12
13 hotelDataHandler = DataHandler(HOTEL_FILE_PATH)
14 hotelInstance = Hotel(hotelDataHandler)
15 customerDataHandler = DataHandler(CUSTOMER_FILE_PATH)
16 customerInstance = Customer(customerDataHandler)
17
18
19 createdHotel1 = hotelInstance.create_hotel(
20     "name", "country", "state", "address"
21 )
22 createdHotel2 = hotelInstance.create_hotel(
23     "name2", "country2", "state2", "address2"
24 )
25 hotelInstance.delete_hotel(2)
26 hotelToModify = createdHotel2[0]
27 hotelToModify["name"] = "modifiedName"
28 modifiedHotels = hotelInstance.modify_hotel(hotelToModify)
29 hotelInfo1 = hotelInstance.get_hotel_info(1)
30 print(hotelInfo1)
31 hotelInfo2 = hotelInstance.get_hotel_info(4)
32 print(hotelInfo2)
33
34
35 createdCustomer1 = customerInstance.create_customer("name", "lastName")
36 createdCustomer2 = customerInstance.create_customer("name2", "lastName2")
37 customerList = customerInstance.delete_customer(2)
38 print(createdCustomer2)
39 customerToModify = customerList[0]
40 customerToModify["name"] = "modifiedName"
41 modifiedCustomers = customerInstance.modify_customer(customerToModify)
42 customerInfo1 = customerInstance.get_customer_data(1)
43 print(customerInfo1)
44 customerInfo2 = customerInstance.get_customer_data(5)
45 print(customerInfo2)
46
47 reservation = hotelInstance.make_reservation(
48     customerInfo1["id"], hotelInfo1["id"], "23/12/2025", "28/12/2025"
49 )
50 reservation = hotelInstance.make_reservation(
51     5, hotelInfo1["id"], "23/12/2025", "28/12/2025"
52 )
53 print(reservation)
54
55 reservations = hotelInstance.cancel_reservation(1)
56 print(reservations)
57 reservations = hotelInstance.cancel_reservation(4)
58 print(reservations)
59
```

Pruebas unitarias clase hotel

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py X  reservation_test.py

test > unit > classes > hotel_test.py > TestHotel > setUp

1  '''Archivo para probar la clase Hotel'''
2
3
4  import unittest
5
6  from unittest.mock import patch, call
7  from app.classes.data_handler import DataHandler
8  from app.classes.hotel import Hotel
9  from app.classes.reservation import Reservation
10
11
12  class TestHotel(unittest.TestCase):
13      '''Calse para probar la clase Hotel'''
14
15      def setUp(self):
16          data_handler = DataHandler("test.json")
17          self.hotel_instance = Hotel(data_handler)
18          self.test_hotel = [{
19              "name": "hotel",
20              "country": "country",
21              "state": "state",
22              "address": "address",
23              "id": 1
24          }]
25          self.test_reservation = [{
26              "hotel_id": "hotel_id",
27              "customer_id": "customer_id",
28              "check_in_date": "check_in_date",
29              "check_out_date": "check_out_date",
30              "Status": "confirmed",
31              "id": 1
32          }]
33
34      # Test create Hotel
35      @patch.object(DataHandler, 'is_missing', return_value=False)
36      @patch.object(DataHandler, 'create_registry', return_value=[{
37          "name": "hotel",
38          "country": "country",
39          "state": "state",
40          "address": "address",
41          "id": 1
42      }])
43      def test_create_hotel_sucesfull(
44          self, mock_create_registry, mock_is_missing
45      ):
46          '''Metodo para verificar
47          el correcto
48          funcionamiento de crear hotel'''
49          hotel = self.test_hotel[0].copy()
50          del hotel["id"]
51          result = self.hotel_instance.create_hotel(
52              hotel["name"],
53              hotel["country"],
54              hotel["state"],
55              hotel["address"]
56          )
57          mock_is_missing.assert_has_calls([
58              call(hotel["name"]),
59              call(hotel["country"]),
60              call(hotel["state"]),
61              call(hotel["address"])
62          ], any_order=True)
63          mock_create_registry.assert_called_once_with(hotel)
64          self.assertEqual(result, self.test_hotel)
```

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py X  reservat

test > unit > classes > hotel_test.py > TestHotel > setUp
12  class TestHotel(unittest.TestCase):
43      def test_create_hotel_sucesfull(
62          ], any_order=True)
63          mock_create_registry.assert_called_once_with(hotel)
64          self.assertEqual(result, self.test_hotel)
65
66      @patch.object(DataHandler, 'is_missing', return_value=True)
67      def test_create_hotel_missing_value(self, mock_is_missing):
68          '''Metodo para verificar que se levante la excpion
69          correcta al tener información faltante'''
70          hotel = self.test_hotel[0].copy()
71          result = self.hotel_instance.create_hotel(
72              hotel["name"],
73              hotel["country"],
74              hotel["state"],
75              hotel["address"]
76          )
77          del hotel["id"]
78          mock_is_missing.assert_called()
79          self.assertRaises(ValueError)
80          self.assertEqual(result, "Missing data")
81
82      # Test Delete Hotele
83
84      @patch.object(DataHandler, 'is_missing', return_value=False)
85      @patch.object(DataHandler, 'delete_from_file', return_value=[])
86      def test_delete_hotel_sucesfull(
87          self,
88          mock_delete_from_file,
89          mock_is_missing
90      ):
91          '''Metodo para verificar
92          el correcto
93          funcionamiento del metodo delete hotel'''
94          id_to_delete = self.test_hotel[0]["id"]
95          deleted_hotel = self.test_hotel.copy()
96          del deleted_hotel[0]
97          result = self.hotel_instance.delete_hotel(id_to_delete)
98          mock_is_missing.assert_called_once_with(id_to_delete)
99          mock_delete_from_file.assert_called_once_with(id_to_delete)
100         self.assertEqual(result, deleted_hotel)
101
102         @patch.object(DataHandler, 'is_missing', return_value=True)
103         def test_delete_hotel_missing_value(self, mock_is_missing):
104             '''Metodo para verificar que se levante la excpion
105             correcta al tener información faltante'''
106             id_to_delete = self.test_hotel[0]["id"]
107             result = self.hotel_instance.delete_hotel(id_to_delete)
108             mock_is_missing.assert_called_once_with(id_to_delete)
109             self.assertRaises(ValueError)
110             self.assertEqual(result, "Missing data")
111
```

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py
test > unit > classes > hotel_test.py > TestHotel > setUp
12  class TestHotel(unittest.TestCase):
111
112     # Test get hotel info
113     @patch.object(DataHandler, 'is_missing', return_value=False)
114     @patch.object(DataHandler, 'get_data_by_id', return_value={
115         "name": "hotel",
116         "country": "country",
117         "state": "state",
118         "address": "address",
119         "id": 1
120     })
121     def test_get_hotel_info_sucesfull(
122         self, mock_get_data_by_id, mock_is_missing
123     ):
124         '''Metodo para verificar
125         el correcto
126         funcionamiento del metodo get hotel info'''
127         expected_info = self.test_hotel[0]
128         id_to_get = expected_info["id"]
129         result = self.hotel_instance.get_hotel_info(id_to_get)
130         mock_is_missing.assert_called_once_with(id_to_get)
131         mock_get_data_by_id.assert_called_once_with(id_to_get)
132         self.assertEqual(result, expected_info)
133
134     @patch.object(DataHandler, 'is_missing', return_value=True)
135     def test_get_hotel_info_missing_value(self, mock_is_missing):
136         '''Metodo para verificar que se levante la expcion
137         correcta al tener información faltante'''
138         id_to_get = self.test_hotel[0]["id"]
139         result = self.hotel_instance.get_hotel_info(id_to_get)
140         mock_is_missing.assert_called_once_with(id_to_get)
141         self.assertRaises(ValueError)
142         self.assertEqual(result, "Missing data")
143
144     # Test Modify hotel
145
146     @patch.object(DataHandler, 'is_missing', return_value=False)
147     @patch.object(DataHandler, 'modify_file', return_value=[{
148         "name": "modified Name",
149         "country": "country",
150         "state": "state",
151         "address": "address",
152         "id": 1
153     }])
154     def test_modify_hotel_sucesfull(self, mock_modify_file, mock_is_missing):
155         '''Metodo para verificar
156         el correcto
157         funcionamiento del metodo modify hotel'''
158         hotel_to_modify = self.test_hotel[0].copy()
159         hotel_to_modify["name"] = "modified Name"
160         result = self.hotel_instance.modify_hotel(hotel_to_modify)
161         mock_is_missing.assert_has_calls([
162             call(hotel_to_modify["name"]),
163             call(hotel_to_modify["country"]),
164             call(hotel_to_modify["state"]),
165             call(hotel_to_modify["address"]),
166             call(hotel_to_modify["id"])
167         ], any_order=True)
168         mock_modify_file.assert_called_once_with(hotel_to_modify)
169         self.assertEqual(result, [hotel_to_modify])
```

```

@patch.object(DataHandler, 'is_missing', return_value=True)
def test_modify_hotel_missing_value(self, mock_is_missing):
    '''Metodo para verificar que se levante la expcion
    correcta al tener información faltante'''
    hotel_to_modify = self.test_hotel[0].copy()
    hotel_to_modify["name"] = "modified Name"
    result = self.hotel_instance.modify_hotel(hotel_to_modify)
    mock_is_missing.assert_called_once()
    self.assertRaises(ValueError)
    self.assertEqual(result, "Missing data")

# Test Create reservation
@patch.object(Reservation, 'create_reservation', return_value=[{
    "hotel_id": "hotel_id",
    "customer_id": "customer_id",
    "check_in_date": "check_in_date",
    "check_out_date": "check_out_date",
    "Status": "confirmed",
    "id": 1
}]))
def test_make_reservation_sucesfull(self, mock_create_reservation):
    '''Metodo para verificar el
    comportamiento correcto
    del metodo make reservation'''
    reservation_to_make = self.test_reservation[0].copy()
    del reservation_to_make["Status"]
    del reservation_to_make["id"]
    result = self.hotel_instance.make_reservation(
        reservation_to_make["customer_id"],
        reservation_to_make["hotel_id"],
        reservation_to_make["check_in_date"],
        reservation_to_make["check_out_date"]
    )
    mock_create_reservation.assert_called_once_with(
        reservation_to_make,
        self.hotel_instance
    )
    self.assertEqual(result, self.test_reservation)

@patch.object(
    Reservation,
    'create_reservation',
    return_value=("Hotel or Customer with specified ID does not exist")
)
def test_make_reservation_no_id_found(self, mock_create_reservation):
    '''Metodo para verificar que arroja
    ell error correcto cuando no se encuentra
    el hotel o el customer'''
    reservation_to_make = self.test_reservation[0].copy()
    del reservation_to_make["Status"]
    del reservation_to_make["id"]
    result = self.hotel_instance.make_reservation(
        reservation_to_make["customer_id"],
        reservation_to_make["hotel_id"],
        reservation_to_make["check_in_date"],
        reservation_to_make["check_out_date"]
    )
    mock_create_reservation.assert_called_once_with(
        reservation_to_make,
        self.hotel_instance
    )
    self.assertRaises(ValueError)
    self.assertEqual(result, "Failed creating reservation")

```

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py X  reserva

test > unit > classes > hotel_test.py > TestHotel > setUp
12  class TestHotel(unittest.TestCase):
213      return_value=( hotel or customer with specified id does not exist )
214  )
215  def test_make_reservation_no_id_found(self, mock_create_reservation):
216      '''Metodo para verificar que arroja
217      ell error correcto cuando no se encuentra
218      el hotel o el customer'''
219      reservation_to_make = self.test_reservation[0].copy()
220      del reservation_to_make["Status"]
221      del reservation_to_make["id"]
222      result = self.hotel_instance.make_reservation(
223          reservation_to_make["customer_id"],
224          reservation_to_make["hotel_id"],
225          reservation_to_make["check_in_date"],
226          reservation_to_make["check_out_date"]
227      )
228      mock_create_reservation.assert_called_once_with(
229          reservation_to_make,
230          self.hotel_instance
231      )
232      self.assertRaises(ValueError)
233      self.assertEqual(result, "Failed creating reservation")
234
235  # Test cancel_reservation
236  @patch.object(Reservation, 'cancel_reservation', return_value=[{
237      "hotel_id": "hotel_id",
238      "customer_id": "customer_id",
239      "check_in_date": "check_in_date",
240      "check_out_date": "check_out_date",
241      "Status": "canceled",
242      "id": 1
243  }])
244  def test_cancel_reservation_succesfull(self, mock_cancel_reservation):
245      '''Metodo para verificar
246      el correcto funcionamiento
247      del metodo cancel reservation
248      '''
249      reservation_to_cancel = self.test_reservation[0].copy()
250      id_to_cancel = reservation_to_cancel["id"]
251      result = self.hotel_instance.cancel_reservation(id_to_cancel)
252      mock_cancel_reservation.assert_called_once_with(id_to_cancel)
253      reservation_to_cancel["Status"] = "canceled"
254      self.assertEqual(result, [reservation_to_cancel])
255
256
257  if __name__ == "__main__":
258      unittest.main()
259
```

Pruebas Unitarias Clase Customer

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py  data_handler_test.py  customer_test.py X  hotels.json M

test > unit > classes > customer_test.py > TestCustomer > setUp

1  '''Archivo para probar la clase Hotel'''
2
3
4  import unittest
5
6  from unittest.mock import patch, call
7  from app.classes.data_handler import DataHandler
8  from app.classes.customer import Customer
9
10
11  class TestCustomer(unittest.TestCase):
12      '''Calse para probar la clase Customer'''
13
14      def setUp(self):
15          data_handler = DataHandler("test.json")
16          self.customer_instance = Customer(data_handler)
17          self.test_customer = [{
18              "name": "customer",
19              "lastName": "lastName",
20              "id": 1
21          }]
22
23      # Test create customer
24      @patch.object(DataHandler, 'is_missing', return_value=False)
25      @patch.object(DataHandler, 'create_registry', return_value=([
26          "name": "customer",
27          "lastName": "lastName",
28          "id": 1
29      ]))
30      def test_create_customer_sucesfull(
31          self,
32          mock_create_registry,
33          mock_is_missing
34      ):
35          '''Metodo para verificar
36          el correcto
37          funcionamiento de crean Customer'''
38          customer = self.test_customer[0].copy()
39          del customer["id"]
40          result = self.customer_instance.create_customer(
41              customer["name"],
42              customer["lastName"]
43          )
44          mock_is_missing.assert_has_calls([
45              call(customer["name"]),
46              call(customer["lastName"]),
47          ], any_order=True)
48          mock_create_registry.assert_called_once_with(customer)
49          self.assertEqual(result, self.test_customer)
50
51      @patch.object(DataHandler, 'is_missing', return_value=True)
52      def test_create_customer_missing_value(self, mock_is_missing):
53          '''Metodo para verificar que se levante la expcion
54          correcta al tener informacion faltante'''
55          customer = self.test_customer[0].copy()
56          del customer["id"]
57          result = self.customer_instance.create_customer(
58              customer["name"],
59              customer["lastName"]
60          )
61          mock_is_missing.assert_called()
62          self.assertRaises(ValueError)
63          self.assertEqual(result, "Missing data")
64
```



```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py  data_handler_test.py  customer_test.py x  hotels.json M  ()

test > unit > classes > customer_test.py > TestCustomer > setUp
11 class TestCustomer(unittest.TestCase):
12     @patch.object(DataHandler, 'is_missing', return_value=False)
13     @patch.object(DataHandler, 'delete_from_file', return_value=[])
14     # Test delete_customer
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67 @patch.object(DataHandler, 'is_missing', return_value=False)
68 @patch.object(DataHandler, 'delete_from_file', return_value=[])
69 def test_delete_customer_successfull(
70     self,
71     mock_delete_from_file,
72     mock_is_missing
73 ):
74     '''Metodo para verificar
75     el correcto
76     funcionamiento del metodo delete hotel'''
77     id_to_delete = self.test_customer[0]["id"]
78     deleted_customer = self.test_customer.copy()
79     del deleted_customer[0]
80     result = self.customer_instance.delete_customer(id_to_delete)
81     mock_is_missing.assert_called_once_with(id_to_delete)
82     mock_delete_from_file.assert_called_once_with(id_to_delete)
83     self.assertEqual(result, deleted_customer)
84
85 @patch.object(DataHandler, 'is_missing', return_value=True)
86 def test_delete_customer_missing_value(self, mock_is_missing):
87     '''Metodo para verificar que se levante la expcion
88     correcta al tener informacion faltante'''
89     id_to_delete = self.test_customer[0]["id"]
90     result = self.customer_instance.delete_customer(id_to_delete)
91     mock_is_missing.assert_called_once_with(id_to_delete)
92     self.assertRaises(ValueError)
93     self.assertEqual(result, "Missing data")
94
95 # Test modify_customer
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```
hotel.py  main.py M X  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py  data_handler_test.py  customer_test.py X  {} hotel

test > unit > classes > customer_test.py > TestCustomer > setUp
11 class TestCustomer(unittest.TestCase):
124
122 @patch.object(DataHandler, 'is_missing', return_value=True)
123 def test_modify_customer_missing_value(self, mock_is_missing):
124     '''Metodo para verificar que se levante la expcion
125     correcta al tener información faltante'''
126     customer_to_modify = self.test_customer[0].copy()
127     customer_to_modify["name"] = "modified Name"
128     result = self.customer_instance.modify_customer(customer_to_modify)
129     mock_is_missing.assert_called_once()
130     self.assertRaises(ValueError)
131     self.assertEqual(result, "Missing data")
132
133 # Test get customer data
134 @patch.object(DataHandler, 'is_missing', return_value=False)
135 @patch.object(DataHandler, 'get_data_by_id', return_value={
136     "name": "customer",
137     "lastName": "lastName",
138     "id": 1
139 })
140 def test_get_customer_data_sucesfull(
141     self, mock_get_data_by_id, mock_is_missing
142 ):
143     '''Metodo para verificar
144     el correcto
145     funcionamiento del metodo get customer data'''
146     expected_info = self.test_customer[0]
147     id_to_get = expected_info["id"]
148     result = self.customer_instance.get_customer_data(id_to_get)
149     mock_is_missing.assert_called_once_with(id_to_get)
150     mock_get_data_by_id.assert_called_once_with(id_to_get)
151     self.assertEqual(result, expected_info)
152
153 @patch.object(DataHandler, 'is_missing', return_value=True)
154 def test_get_customer_data_missing_value(self, mock_is_missing):
155     '''Metodo para verificar que se levante la expcion
156     correcta al tener información faltante'''
157     id_to_get = self.test_customer[0]["id"]
158     result = self.customer_instance.get_customer_data(id_to_get)
159     mock_is_missing.assert_called_once_with(id_to_get)
160     self.assertRaises(ValueError)
161     self.assertEqual(result, "Missing data")
162
163
164 if __name__ == "__main__":
165     unittest.main()
166
```

Pruebas unitarias clase reservation

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py X  data_handler_test.py  cus

test > unit > classes > reservation_test.py > TestReservation > test_create_reservation_sucesfull

1 '''Archivo para probar la clase Reservation'''
2
3
4 import unittest
5
6 from unittest.mock import patch, call
7 from app.classes.customer import Customer
8 from app.classes.data_handler import DataHandler
9 from app.classes.reservation import Reservation
10 from app.classes.hotel import Hotel
11
12
13 class TestReservation(unittest.TestCase):
14     '''Calse para probar la clase Reservation'''
15
16     def setUp(self):
17         data_handler = DataHandler("test.json")
18         self.reservation_instance = Reservation(data_handler)
19         self.customer_instance = Customer(data_handler)
20         self.hotel_instance = Hotel(data_handler)
21         self.test_reservation = [{
22             "hotelID": 1,
23             "CustomerID": 1,
24             "checkInDate": "check_in_date",
25             "checkOutDate": "check_out_date",
26             "Status": "confirmed",
27             "id": 1
28         }]
29
```

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py x  data_handler_test.py  custo

test > unit > classes > reservation_test.py > TestReservation > test_create_reservation_sucesfull
13 class TestReservation(unittest.TestCase):
14     def setUp(self):
15
16
17
18     ]]
19
20
21     # test create_reservation
22     @patch.object(DataHandler, 'is_missing', return_value=False)
23     @patch.object(Customer, 'get_customer_data', return_value={
24         "name": "name",
25         "lastName": "lastName",
26         "id": 1
27     })
28     @patch.object(Hotel, 'get_hotel_info', return_value={
29         "name": "name",
30         "country": "country",
31         "state": "state",
32         "address": "address",
33         "id": 1
34     })
35     @patch.object(DataHandler, 'create_registry', return_value=[
36         "hotelID": 1,
37         "CustomerID": 1,
38         "checkInDate": "check_in_date",
39         "checkOutDate": "check_out_date",
40         "Status": "confirmed",
41         "id": 1
42     ])
43     def test_create_reservation_sucesfull(
44         self,
45         mock_create_registry,
46         mock_get_hotel_info,
47         mock_get_customer_data,
48         mock_is_missing
49     ):
50         '''Metodo para verificar
51         el correcto
52         funcionamiento de crear reservación'''
53         hotel_reservation = {
54             "hotel_id": 1,
55             "customer_id": 1,
56             "check_in_date": "check_in_date",
57             "check_out_date": "check_out_date",
58             "Status": "confirmed"
59         }
60         expected_reservation = self.test_reservation[0]
61         reservation = expected_reservation.copy()
62         del reservation["id"]
63         result = self.reservation_instance.create_reservation(
64             hotel_reservation,
65             self.hotel_instance
66         )
67         mock_is_missing.assert_has_calls([
68             call(hotel_reservation["hotel_id"]),
69             call(hotel_reservation["customer_id"]),
70             call(hotel_reservation["check_in_date"]),
71             call(hotel_reservation["check_out_date"]),
72         ])
73         mock_get_customer_data.assert_called_once_with(
74             hotel_reservation["customer_id"]
75         )
76         mock_get_hotel_info.assert_called_once_with(
77             hotel_reservation["hotel_id"]
78         )
79         print(expected_reservation)
80         mock_create_registry.assert_called_once_with(reservation)
81         self.assertEqual(result, [expected_reservation])
82
83
84
85
86
87
88
89
90
91
```

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py X  data_handler_test.py  customer_test.py

test > unit > classes > reservation_test.py > TestReservation > test_create_reservation_sucesfull
13 class TestReservation(unittest.TestCase):
90     self.assertEqual(result, [expected_reservation])
91
92     @patch.object(DataHandler, 'is_missing', return_value=True)
93     def test_create_reservation_missing_value(self, mock_is_missing):
94         '''Metodo para verificar que se levante la expcion
95         correcta al tener información faltante'''
96         hotel_reservation = {
97             "hotel_id": 1,
98             "customer_id": 1,
99             "check_in_date": "",
100             "check_out_date": "check_out_date",
101             "Status": "confirmed"
102         }
103         result = self.reservation_instance.create_reservation(
104             hotel_reservation,
105             self.hotel_instance
106         )
107         mock_is_missing.assert_called_once()
108         self.assertRaises(ValueError)
109         self.assertEqual(result, "Missing data")
110
111     @patch.object(DataHandler, 'is_missing', return_value=False)
112     @patch.object(Customer, 'get_customer_data', return_value="No data found")
113     @patch.object(Hotel, 'get_hotel_info', return_value=[{
114         "name": "name",
115         "country": "country",
116         "state": "state",
117         "address": "address",
118         "id": 1
119     }])
120     def test_create_reservation_customer_not_found(
121         self,
122         mock_get_hotel_info,
123         mock_get_customer_data,
124         mock_is_missing
125     ):
126         '''Metodo para validar
127         que arroja el error correcto
128         cuando no encuentra el customer'''
129         hotel_reservation = {
130             "hotel_id": 2,
131             "customer_id": 2,
132             "check_in_date": "check_in_date",
133             "check_out_date": "check_out_date",
134             "Status": "confirmed"
135         }
136         expected_reservation = self.test_reservation[0]
137         reservation = expected_reservation.copy()
138         del reservation["id"]
139         result = self.reservation_instance.create_reservation(
140             hotel_reservation,
141             self.hotel_instance
142         )
143         mock_is_missing.assert_has_calls([
144             call(hotel_reservation["hotel_id"]),
145             call(hotel_reservation["customer_id"]),
146             call(hotel_reservation["check_in_date"]),
147             call(hotel_reservation["check_out_date"]),
148         ])
149         mock_get_customer_data.assert_called_once_with(
150             hotel_reservation["customer_id"]
151         )
152         mock_get_hotel_info.assert_called_once_with(
153             hotel_reservation["hotel_id"]
154         )
155         self.assertEqual(
```

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py X  data_handler_test.py

test > unit > classes > reservation_test.py > TestReservation > test_create_reservation_sucesfull
13  class TestReservation(unittest.TestCase):
120 def test_create_reservation_customer_not_found(
154     )
155     self.assertEqual(
156         result,
157         "Hotel or Customer with specified ID does not exist"
158     )
159
160     @patch.object(DataHandler, 'is_missing', return_value=False)
161     @patch.object(Customer, 'get_customer_data', return_value={
162         "name": "name",
163         "lastName": "lastName",
164         "id": 1
165     })
166     @patch.object(Hotel, 'get_hotel_info', return_value="No data found")
167     def test_create_reservation_hotel_not_found(
168         self,
169         mock_get_hotel_info,
170         mock_get_customer_data,
171         mock_is_missing
172     ):
173         '''Metodo para validar
174         que arroja el error correcto
175         cuando no encuentra el hotel'''
176         hotel_reservation = {
177             "hotel_id": 2,
178             "customer_id": 2,
179             "check_in_date": "check_in_date",
180             "check_out_date": "check_out_date",
181             "Status": "confirmed"
182         }
183         expected_reservation = self.test_reservation[0]
184         reservation = expected_reservation.copy()
185         del reservation["id"]
186         result = self.reservation_instance.create_reservation(
187             hotel_reservation,
188             self.hotel_instance
189         )
190         mock_is_missing.assert_has_calls([
191             call(hotel_reservation["hotel_id"]),
192             call(hotel_reservation["customer_id"]),
193             call(hotel_reservation["check_in_date"]),
194             call(hotel_reservation["check_out_date"]),
195         ])
196         mock_get_customer_data.assert_called_once_with(
197             hotel_reservation["customer_id"]
198         )
199         mock_get_hotel_info.assert_called_once_with(
200             hotel_reservation["hotel_id"]
201         )
202         self.assertEqual(
203             result,
204             "Hotel or Customer with specified ID does not exist"
205         )
206
```

hotel.py main.py M customer.py reservation.py data_handler.py hotel_test.py

test > unit > classes > reservation_test.py > TestReservation > test_create_reservation_sucesfull

```
13 class TestReservation(unittest.TestCase):
14     ,
15
16
17
18
19
20
21
22
23
24
25
26
27     # test cancel_reservation
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py X  data_handler_test.py

test > unit > classes > reservation_test.py > TestReservation > test_create_reservation_sucesfull
13  class TestReservation(unittest.TestCase):

261
262     # get_reservation_by_id
263
264     @patch.object(DataHandler, 'is_missing', return_value=False)
265     @patch.object(DataHandler, 'get_data_by_id', return_value={
266         "hotelID": 1,
267         "CustomerID": 1,
268         "checkInDate": "check_in_date",
269         "checkOutDate": "check_out_date",
270         "Status": "confirmed",
271         "id": 1
272     })
273     def test_get_reservation_by_id_sucesfull(
274         self, mock_get_data_by_id, mock_is_missing
275     ):
276         '''Metodo para verificar
277         el correcto
278         funcionamiento del metodo get reservation'''
279         expected_info = self.test_reservation[0]
280         id_to_get = expected_info["id"]
281         result = self.reservation_instance.get_reservation_by_id(id_to_get)
282         mock_is_missing.assert_called_once_with(id_to_get)
283         mock_get_data_by_id.assert_called_once_with(id_to_get)
284         self.assertEqual(result, expected_info)
285
286     @patch.object(DataHandler, 'is_missing', return_value=True)
287     def test_get_reservation_by_missing_value(
288         self,
289         mock_is_missing
290     ):
291         '''Metodo para verificar que se levante la expcion
292         correcta al tener información faltante'''
293         id_reservation = self.test_reservation[0]["id"]
294         result = self.reservation_instance.get_reservation_by_id(
295             id_reservation
296         )
297         mock_is_missing.assert_called_once_with(id_reservation)
298         self.assertRaises(ValueError)
299         self.assertEqual(result, "Missing data")
300
301
302 if __name__ == "__main__":
303     unittest.main()
304
```

Pruebas unitarias clase datahandler

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py  data_handler_test.py X  customer

test > unit > classes > data_handler_test.py > ...

1  '''Archivo para probar la clase DataHandler'''
2
3
4  import unittest
5  import json
6
7  from unittest.mock import mock_open, patch
8  from app.classes.data_handler import DataHandler
9
10 # pylint: disable=too-many-public-methods
11 class TestDataHandler(unittest.TestCase):
12     '''Calse para probar la clase DataHandler'''
13
14     def setUp(self):
15         self.data_handler = DataHandler("test.json")
16         self.new_data = {"name": "Hernández"}
17         self.data_to_modify = {"name": "Hernández", "id": 2}
18         self.test_data = [
19             {"name": "Andre", "id": 1},
20             {"name": "Maximiliano", "id": 2}
21         ]
22
23     # Test create_registry
24     @patch.object(DataHandler, 'file_as_json', return_value=[
25         {"name": "Andre", "id": 1},
26         {"name": "Maximiliano", "id": 2}
27     ])
28     @patch.object(DataHandler, 'write_to_file', return_value=[
29         {"name": "Andre", "id": 1},
30         {"name": "Maximiliano", "id": 2},
31         {"name": "Hernández", "id": 3}
32     ])
33     @patch.object(DataHandler, 'create_id', return_value=3)
34     def test_create_registry_success(
35         self, mock_create_id, mock_write_to_file, mock_file_as_json
36     ):
37         """Test que se agregue exitosamente a un archivo"""
38         result = self.data_handler.create_registry(self.new_data)
39         expected_file_content = [
40             self.test_data[0],
41             self.test_data[1],
42             self.new_data
43         ]
44         mock_file_as_json.assert_called_once()
45         mock_create_id.assert_called_once_with(expected_file_content)
46         mock_write_to_file.assert_called_once_with(expected_file_content)
47         self.assertEqual(result, expected_file_content)
48
49     @patch.object(DataHandler, 'file_as_json', return_value=[])
50     @patch.object(DataHandler, 'write_to_file', return_value=[
51         {"name": "Hernández", "id": 1}
52     ])
53     @patch.object(DataHandler, 'create_id', return_value=1)
54     def test_create_registry_empty_file(
55         self, mock_create_id, mock_write_to_file, mock_file_as_json
56     ):
57         """Test agregar un registro cuando el archivo no tiene un registro"""
58         result = self.data_handler.create_registry(self.new_data)
59         expected_file_content = [self.new_data]
60         mock_file_as_json.assert_called_once()
61         mock_create_id.assert_called_once()
62         mock_write_to_file.assert_called_once_with(expected_file_content)
63         self.assertEqual(result, expected_file_content)
64
```



```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py  data_handler_test.py > ...
test > unit > classes > data_handler_test.py > ...
11 class TestDataHandler(unittest.TestCase):
12     self.assertEqual(result, expected_file_content)
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65 # Test modify_file
66 @patch.object(DataHandler, 'file_as_json', return_value=[
67     {"name": "Andre", "id": 1},
68     {"name": "Maximiliano", "id": 2}
69 ])
70 @patch.object(DataHandler, 'find_index_of_array', return_value=1)
71 @patch.object(DataHandler, 'write_to_file', return_value=[
72     {"name": "Andre", "id": 1},
73     {"name": "Hernández", "id": 2}
74 ])
75 def test_modify_file_found_index(
76     self, mock_write_to_file, mock_find_index_of_array, mock_file_as_json
77 ):
78     """Test modificar un registro con id encontrado"""
79     result = self.data_handler.modify_file(self.data_to_modify)
80     expected_file_content = [self.test_data[0], self.data_to_modify]
81     expected_id = 2
82     mock_file_as_json.assert_called_once()
83     mock_find_index_of_array.assert_called_once_with(
84         expected_file_content, expected_id
85     )
86     mock_write_to_file.assert_called_once_with(
87         expected_file_content
88     )
89     self.assertEqual(result, expected_file_content)
90
91 @patch.object(DataHandler, 'file_as_json', return_value=[
92     {"name": "Andre", "id": 1}
93 ])
94 @patch.object(DataHandler, 'find_index_of_array', return_value=-1)
95 def test_modify_file_not_found(
96     self, mock_find_index_of_array, mock_file_as_json
97 ):
98     """Test no modificar un registro si el id no es encontrado"""
99     result = self.data_handler.modify_file(self.data_to_modify)
100     expected_file_content = [self.test_data[0]]
101     expected_id = 2
102     mock_file_as_json.assert_called_once()
103     mock_find_index_of_array.assert_called_once_with(
104         expected_file_content, expected_id
105     )
106     self.assertEqual(result, expected_file_content)
107
108
```

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py  data_ha

test > unit > classes > data_handler_test.py > ...
11  class TestDataHandler(unittest.TestCase):
12  def test_get_data_by_id_found(self):
106     self.assertEqual(result, expected_file_content)
107
108     # Test get Data by id
109
110     @patch.object(DataHandler, 'file_as_json', return_value=[
111         {"name": "Andre", "id": 1},
112         {"name": "Maximiliano", "id": 2}
113     ])
114     @patch.object(DataHandler, 'find_index_of_array', return_value=1)
115     def test_get_data_by_id_found(
116         self, mock_find_index_of_array, mock_file_as_json
117     ):
118         """Test que entrega la data del archivo que tenga el id"""
119         result = self.data_handler.get_data_by_id(2)
120         expected_data = self.test_data[1]
121         expected_id = 2
122         mock_file_as_json.assert_called_once()
123         mock_find_index_of_array.assert_called_once_with(
124             self.test_data, expected_id
125         )
126         self.assertEqual(result, expected_data)
127
128     @patch.object(DataHandler, 'file_as_json', return_value=[
129         {"name": "Andre", "id": 1},
130         {"name": "Maximiliano", "id": 2}
131     ])
132     @patch.object(DataHandler, 'find_index_of_array', return_value=-1)
133     def test_get_data_by_id_not_found(
134         self, mock_find_index_of_array, mock_file_as_json
135     ):
136         """Test que entregue el mensaje de no data encontrada"""
137         result = self.data_handler.get_data_by_id(3)
138         expected_data = "No data found"
139         expected_id = 3
140         mock_file_as_json.assert_called_once()
141         mock_find_index_of_array.assert_called_once_with(
142             self.test_data, expected_id
143         )
144         self.assertEqual(result, expected_data)
145
146     # Test Delete from File
147     @patch.object(DataHandler, 'file_as_json', return_value=[
148         {"name": "Andre", "id": 1},
149         {"name": "Maximiliano", "id": 2}
150     ])
151     @patch.object(DataHandler, 'delete_from_json', return_value=[
152         {"name": "Andre", "id": 1}
153     ])
154     @patch.object(DataHandler, 'write_to_file', return_value=[
155         {"name": "Andre", "id": 1}
156     ])
157     def test_delete_from_file_success(
158         self, mock_write_to_file, mock_delete_from_json, mock_file_as_json
159     ):
160         """Test que borre del archivo de manera exitosa"""
161         result = self.data_handler.delete_from_file(2)
162         expected_data = [{"name": "Andre", "id": 1}]
163         mock_file_as_json.assert_called_once()
164         mock_delete_from_json.assert_called_once_with(2, self.test_data)
165         mock_write_to_file.assert_called_once_with(expected_data)
166         self.assertEqual(result, expected_data)
```

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py  data_handler_test.py X  custom...
test > unit > classes > data_handler_test.py > ...
11  class TestDataHandler(unittest.TestCase):
166      self.assertEqual(result, expected_data)
167
168      # Test file as json
169      @patch(
170          "builtins.open",
171          new_callable=mock_open,
172          read_data=[
173              '{"name": "Andre", "id": 1}, {"name": "Maximiliano", "id": 2}']
174          )
175
176      def test_file_as_json_success(self, mock_file):
177          """Test que el json leído sea el correcto"""
178          result = self.data_handler.file_as_json()
179          mock_file.assert_called_once()
180          self.assertEqual(result, self.test_data)
181
182      @patch("builtins.open", side_effect=FileNotFoundError)
183      def test_file_as_json_file_not_found(self, mock_file):
184          """Test que maneje de manera correcta el error FileNotFoundError"""
185          result = self.data_handler.file_as_json()
186          mock_file.assert_called_once()
187          self.assertEqual(result, "File Does not exists")
188
189      @patch("builtins.open", new_callable=mock_open, read_data="")
190      def test_file_as_json_invalid_json(self, mock_file):
191          """Test que maneje de manera correcta
192          el error json.decoder.JSONDecodeError"""
193          with self.assertRaises(json.decoder.JSONDecodeError):
194              self.data_handler.file_as_json()
195          mock_file.assert_called_once()
196
197      # Test Filter Json File
198      def test_filter_json_file_successfull(self):
199          """Test de filtrar objetos de un json"""
200          json_to_filter = self.test_data
201          filtered_json = self.data_handler.filter_json_file(1, json_to_filter)
202          self.assertEqual(len(filtered_json), 1)
203          self.assertEqual(filtered_json, [{"name": "Andre", "id": 1}])
204
205      def test_filter_json_file_not_found(self):
206          """Test de filtrar objetos de un json"""
207          json_to_filter = self.test_data
208          filtered_json = self.data_handler.filter_json_file(4, json_to_filter)
209          self.assertEqual(len(filtered_json), 0)
210
211      # Test delete from json
212      def test_delete_from_json_item_deleted(self):
213          """Test que borre el índice dl id del json"""
214          json_to_delete = self.test_data
215          id_to_delete = 2
216          deleted_json = self.data_handler.delete_from_json(
217              id_to_delete, json_to_delete
218          )
219          self.assertNotIn(
220              {"name": "Maximiliano", "id": 2}, deleted_json
221          )
222
223
```

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py  data_handler_test.py X
test > unit > classes > data_handler_test.py > ...
11 class TestDataHandler(unittest.TestCase):
221     )
222
223     # Test find index of Array
224     def test_find_index_of_array_found_index(self):
225         """Test que encuentre el índice del id proporcionado"""
226         json_to_filter = self.test_data
227         index = self.data_handler.find_index_of_array(json_to_filter, 1)
228         self.assertEqual(index, 0)
229
230     def test_find_index_of_array_not_found_index(self):
231         """Test que regrese -1 cuando no encuentra el id"""
232         json_to_filter = self.test_data
233         index = self.data_handler.find_index_of_array(json_to_filter, 4)
234         self.assertEqual(index, -1)
235
236     # Test write to file
237     @patch(
238         "builtins.open",
239         new_callable=mock_open,
240         read_data=(
241             '[{"name": "Andre", "id": 1}, {"name": "Maximiliano", "id": 2}]'
242         )
243     )
244     def test_write_to_file_success(self, mock_file):
245         """Test de que se escribe de manera correcta al archivo"""
246         result = self.data_handler.write_to_file(self.test_data)
247         mock_file.assert_called_with("test.json", "w", encoding="utf-8")
248         self.assertEqual(result, self.test_data)
249
250     @patch("builtins.open", side_effect=TypeError)
251     def test_write_to_file_failure(self, mock_file):
252         """Test de write_to_file maneja errores"""
253         result = self.data_handler.write_to_file(self.test_data)
254         mock_file.assert_called_once()
255         self.assertRaises(TypeError)
256         self.assertEqual(result, "Error writing to file")
257
258     # Test create ID
259     def test_create_id_successfull(self):
260         """Metodo to test creation of id"""
261         result = self.data_handler.create_id(self.test_data)
262         self.assertEqual(result, 3)
263
264     def test_create_id_empty(self):
265         """Metodo to test creation of id when file is empty"""
266         result = self.data_handler.create_id([])
267         self.assertEqual(result, 1)
268
```

```
hotel.py  main.py M  customer.py  reservation.py  data_handler.py  hotel_test.py  reservation_test.py  data_handler_test.py X
test > unit > classes > data_handler_test.py > ...
11 class TestDataHandler(unittest.TestCase):
264     def test_create_id_empty(self):
267         self.assertEqual(result, 1)
268
269     # Test is missing
270     def test_is_missing_no_missing(self):
271         """Metodo para verificar que regresa
272         false si el valor no es nulo o vacío"""
273         result = self.data_handler.is_missing("test")
274         self.assertEqual(result, False)
275
276     def test_is_missing_none(self):
277         """Metodo para verificar que regresa
278         true si el valor es nulo"""
279         result = self.data_handler.is_missing(None)
280         self.assertEqual(result, True)
281
282     def test_is_missing_empty(self):
283         """Metodo para verificar que regresa
284         true si el valor es vacío"""
285         result = self.data_handler.is_missing("")
286         self.assertEqual(result, True)
287
288
289 if __name__ == "__main__":
290     unittest.main()
291
```

Resultado flake 8

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> flake8 ./
PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> █
```

Resultado Pylint

```
PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> pylint ./
***** Module app.__init__
app\__init__.py:1:0: C0103: Module name "__init__" doesn't conform to snake_case naming style (invalid-name)

-----
Your code has been rated at 9.98/10 (previous run: 9.98/10, +0.00)
```

El error se dejó, ya que ese archivo fue para agregar el correcto formato del modulo y ese debe ser su nombre, no se puede cambiar.

Pruebas Unitarias Sin errores Clase Hotel

```
PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> $env:PYTHONPATH = $PWD
PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> python test/unit/classes/hotel_test.py
.....
-----
Ran 11 tests in 0.004s

OK
PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> █
```

Pruebas unitarias Sin errores Clase Customer

```
PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> python test/unit/classes/customer_test.py
.....
-----
Ran 8 tests in 0.003s

OK
PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> █
```

Pruebas unitarias Sin errores Clase Reservation

```
PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> python test/unit/classes/reservation_test.py
.....{'hotelID': 1, 'CustomerID': 1, 'checkInDate': 'check_in_date', 'checkOutDate': 'check_out_date', 'Status': 'confirmed', 'id': 1}
...
-----
Ran 8 tests in 0.004s

OK
PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> █
```

Pruebas unitarias Sin errores Clase DataHandler

```
PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> python test/unit/classes/data_handler_test.py
.....File does not exists
.....Empty Json, give propper format to file
.....Error writing to file
.....
-----
Ran 22 tests in 0.013s

OK
```

Pruebas Unitarias Forzando un error Clase Hotel

Cambio realizado

```
# Test create Hotel
@patch.object(DataHandler, 'is_missing', return_value=False)
@patch.object(DataHandler, 'create_registry', return_value=[{
    "name": "hotel",
    "country": "country",
    "state": "state",
    "address": "address",
    "id": 1
}])
def test_create_hotel_sucesfull(
    self, mock_create_registry, mock_is_missing
):
    '''Metodo para verificar
    el correcto
    funcionamiento de crear hotel'''
    hotel = self.test_hotel[0].copy()
    del hotel["id"]
    result = self.hotel_instance.create_hotel(
        hotel["name"],
        hotel["country"],
        hotel["state"],
        hotel["address"]
    )
    mock_is_missing.assert_has_calls([
        call(hotel["name"]),
        call(hotel["country"]),
        call(hotel["state"]),
        call(hotel["address"])
    ], any_order=True)
    mock_create_registry.assert_called_once_with(hotel)
    self.assertEqual(result, {})
```

Resultado

```

OK
PS C:\Users\soyel\Documents\Viaestria\Pruebas de Software y aseguramiento de calidad\Semana 6> python test/unit/classes/hotel_test.py
..F.....
=====
FAIL: test_create_hotel_sucesfull (_main_.TestHotel.test_create_hotel_sucesfull)
Metodo para verificar
=====
Traceback (most recent call last):
  File "C:\Users\soyel\AppData\Local\Programs\Python\Python311\Lib\unittest\mock.py", line 1375, in patched
    return func(*newargs, **newkeywargs)
    ~~~~~
  File "C:\Users\soyel\Documents\Viaestria\Pruebas de Software y aseguramiento de calidad\Semana 6\test\unit\classes\hotel_test.py", line 64, in test_create_hotel_sucesfull
    self.assertEqual(result, {})
AssertionError: [{'name': 'hotel', 'country': 'country', [44 chars]: 1}] != {}

-----
Ran 11 tests in 0.008s

FAILED (failures=1)

```

Pruebas unitarias Sin errores Clase Customer

Cambio realizado

```

# Test create customer
@patch.object(DataHandler, 'is_missing', return_value=False)
@patch.object(DataHandler, 'create_registry', return_value=[{
    "name": "customer",
    "lastName": "lastName",
    "id": 1
}])
def test_create_customer_sucesfull(
    self,
    mock_create_registry,
    mock_is_missing
):
    '''Metodo para verificar
    el correcto
    funcionamiento de crear Customer'''
    customer = self.test_customer[0].copy()
    del customer["id"]
    result = self.customer_instance.create_customer(
        customer["name"],
        customer["lastName"]
    )
    mock_is_missing.assert_has_calls([
        call(customer["name"]),
        call(customer["lastName"])
    ], any_order=True)
    mock_create_registry.assert_called_once_with(customer)
    self.assertEqual(result, {})

```

Resultado

```

PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> python test/unit/classes/customer_test.py
.F.....
=====
FAIL: test_create_customer_sucesfull (__main__.TestCustomer.test_create_customer_sucesfull)
Metodo para verificar
-----
Traceback (most recent call last):
  File "C:\Users\soyel\AppData\Local\Programs\Python\Python311\Lib\unittest\mock.py", line 1375, in patched
    return func(*newargs, **newkeywargs)
           ~~~~~^~~~~~
  File "C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6\test\unit\classes\customer_test.py", line 49, in test_create_customer_sucesfull
    self.assertEqual(result, {})
AssertionError: [{'name': 'customer', 'lastName': 'lastName', 'id': 1}] != {}

-----
Ran 8 tests in 0.009s

FAILED (failures=1)

```

Pruebas unitarias Sin errores Clase Reservation

Cambio realizado

```

def test_get_reservation_by_id_sucesfull(
    self, mock_get_data_by_id, mock_is_missing
):
    '''Metodo para verificar
    el correcto
    funcionamiento del metodo get reservation'''
    expected_info = self.test_reservation[0]
    id_to_get = expected_info["id"]
    result = self.reservation_instance.get_reservation_by_id(id_to_get)
    mock_is_missing.assert_called_once_with(id_to_get)
    mock_get_data_by_id.assert_called_once_with(id_to_get)
    self.assertEqual(result, expected_info)

@patch.object(DataHandler, 'is_missing', return_value=True)
def test_get_reservation_by_missing_value(
    self,
    mock_is_missing
):
    '''Metodo para verificar que se levante la expcion
    correcta al tener información faltante'''
    id_reservation = self.test_reservation[0]["id"]
    result = self.reservation_instance.get_reservation_by_id(
        id_reservation
    )
    mock_is_missing.assert_called_once_with(id_reservation)
    self.assertRaises(ValueError)
    self.assertEqual(result, "M")

```

Resultado


```

PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> python test/unit/classes/reservation_test.py
.....{'hotelID': 1, 'CustomerID': 1, 'checkInDate': 'check_in_date', 'checkOutDate': 'check_out_date', 'Status': 'confirmed', 'id': 1}
..F
FAIL: test_get_reservation_by_missing_value (_main_.TestReservation.test_get_reservation_by_missing_value)
Metodo para verificar que se levante la expcion
-----
Traceback (most recent call last):
  File "C:\Users\soyel\AppData\Local\Programs\Python\Python311\Lib\unittest\mock.py", line 1375, in patched
    return func(*newargs, **newkeywargs)
    ~~~~~^~~~~~
  File "C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6\test\unit\classes\reservation_test.py", line 299, in test_get_reservation_by_missing_value
    self.assertEqual(result, "M")
AssertionError: 'Missing data' != 'M'
- Missing data
+ M
-----

Ran 8 tests in 0.007s

FAILED (failures=1)

```

Pruebas unitarias Sin errores Clase DataHandler

Cambio realizado

```

# Test create_registry
@patch.object(DataHandler, 'file_as_json', return_value=[
    {"name": "Andre", "id": 1},
    {"name": "Maximiliano", "id": 2}
])
@patch.object(DataHandler, 'write_to_file', return_value=[
    {"name": "Andre", "id": 1},
    {"name": "Maximiliano", "id": 2},
    {"name": "Hernández", "id": 3}
])
@patch.object(DataHandler, 'create_id', return_value=3)
def test_create_registry_success(
    self, mock_create_id, mock_write_to_file, mock_file_as_json
):
    """Test que se agregue exitosamente a un archivo"""
    result = self.data_handler.create_registry(self.new_data)
    expected_file_content = [
        self.test_data[0],
        self.test_data[1],
        self.new_data
    ]
    mock_file_as_json.assert_called_once()
    mock_create_id.assert_called_once_with(expected_file_content)
    mock_write_to_file.assert_called_once_with(expected_file_content)
    self.assertEqual(result, 1)

```

Resultado

```

PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> python test/unit/classes/data_handler_test.py
...F..File does not exists
.Empty json, give proper format to file
.....Error writing to file
..
=====
FAIL: test_create_registry_success (__main__.TestDataHandler.test_create_registry_success)
Test que se agregue exitosamente a un archivo
-----
Traceback (most recent call last):
  File "C:\Users\soyel\AppData\Local\Programs\Python\Python311\Lib\unittest\mock.py", line 1375, in patched
    return func(*newargs, **newkeywargs)
           ~~~~~^~~~~~
  File "C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6\test\unit\classes\data_handler_test.py", line 48, in test_create_registry_success
    self.assertEqual(result, 1)
AssertionError: [{'name': 'Andre', 'id': 1}, {'name': 'Ma[48 chars]: 3}] != 1

Ran 22 tests in 0.015s

FAILED (failures=1)

```

Corrida programa de control

Archivos de datos antes de la corrida

Terminal Help

hotel.py main.py M {} hotels.json M X {} reservations.json M {} customers.json M

app > data > {} hotels.json

1

Terminal Help

hotel.py main.py M {} hotels.json M {} reservations.json M X {} customers.json M

app > data > {} reservations.json

1

Terminal Help

hotel.py main.py M {} hotels.json M {} reservations.json M {} customers.json M X

app > data > {} customers.json

1

Ejecución programa de control

```

PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> python .\app\main.py
{'name': 'modifiedName', 'country': 'country', 'state': 'state', 'address': 'address', 'id': 1}
No data found
[{'name': 'name', 'lastName': 'lastName', 'id': 1}, {'name': 'name2', 'lastName': 'lastName2', 'id': 2}]
{'name': 'modifiedName', 'lastName': 'lastName', 'id': 1}
No data found
Failed creating reservation
[{'hotelID': 1, 'CustomerID': 1, 'checkInDate': '23/12/2025', 'checkOutDate': '28/12/2025', 'Status': 'canceled', 'id': 1}]
Id does not exist
PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6>

```

Archivos de datos después de la corrida

```

hotel.py  main.py M  {} hotels.json  {} reservations.json M  {} customers.json
app > data > {} hotels.json > {} 0
1 [{"name": "modifiedName", "country": "country", "state": "state", "address": "address", "id": 1}]

```

```

hotel.py  main.py M  {} hotels.json  {} reservations.json M X  {} customers.json
app > data > {} reservations.json > {} 0
1 [{"hotelID": 1, "CustomerID": 1, "checkInDate": "23/12/2025", "checkOutDate": "28/12/2025", "Status": "canceled", "id": 1}]

```

```

hotel.py X  main.py M  {} hotels.json  {} reservations.json M  {} customers.json X
app > data > {} customers.json > ...
1 [{"name": "modifiedName", "lastName": "lastName", "id": 1}]

```

Code Coverage

```

PS C:\Users\soyel\Documents\Maestria\Pruebas de Software y aseguramiento de calidad\Semana 6> coverage report -m
Name                               Stmts  Miss  Cover   Missing
-----
app\classes\__init__.py             0      0  100%
app\classes\customer.py             36      0  100%
app\classes\data_handler.py         72      0  100%
app\classes\hotel.py                55      0  100%
app\classes\reservation.py          38      0  100%
tests\unit\classes\customer_test.py  78      1   99%    165
tests\unit\classes\data_handler_test.py 143      2   99%    196, 291
tests\unit\classes\hotel_test.py    105      1   99%    258
tests\unit\classes\reservation_test.py  95      1   99%    303
-----
TOTAL                               622      5   99%

```

