Manual de Usuario Sistema Hotel Luna Azul

Autores:

Gina Márquez Garzón
Catalina Ortiz Duque
Juliana Herrera Osorio

Facultad de Ingeniería, Universidad de Antioquia
Algoritmia y Programación
Profesor: Víctor Mercado
2025

Tabla de contenido

Introducción

Se desarrollo un proyecto de software que simula el sistema de gestión para el hotel Luna Azul, aplicando los conocimientos adquiridos en el curso de Algoritmia y Programación.

El sistema permite llevar un control básico de huéspedes, reservas, check-out y generación de reportes tanto administrativos como gráficos. Este manual explica la lógica de cada módulo, el uso de librerías, el menú principal y cómo interpretar los resultados.

Librerías:

Se importaron librerías conocidas para cubrir las necesidades del sistema, tales como:

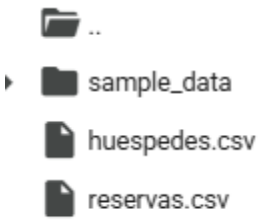
```
import csv
import os
import pandas as pd
from datetime import datetime, timedelta
```

```
import matplotlib.pyplot as plt
```

- La librería **csv** se requiere para instalar y guardar los archivos csv que crea el google colab para almacenar los datos obtenidos del código del sistema.
- La librería **os** es para trabajar con los archivos y las carpetas que crea el sistema operativo, así verifica que un archivo ya existe y evitar duplicar si ya está creado.
- La librería **pandas** analiza los datos en forma de tabla y esto lee y filtra los registros, para agruparlos por documento y sumar las noches o ingresos.
- la from **datetime import datetima, timedelta** sirve para trabajar con las fechas y horas, en este caso, para registrar fecha de ingreso y salida para calcular la cantidad de noches.
- La librería **matplotlib.pyplot as plt** es de utilidad para la creación de gráficos.

Archivos utilizados:

El sistema opera con los siguientes archivos:



Los archivos huéspedes.csv y reservas.csv se guardan automáticamente cuando el sistema registra datos Esto se realiza utilizando el módulo csv de Python, mediante el uso de la función **open()** junto con **csv.writer()**
Los archivos aparecen en el panel lateral izquierdo de Colab y permiten guardar la información en un formato estructurado, fácil de consultar y reutilizar. Esta gestión de archivos hace posible conservar los datos de forma sencilla durante la ejecución del sistema.

- **huéspedes.csv:** Guarda los datos de huéspedes registrados.

- **reservas.csv**: Registra las reservas realizadas con fechas y costos

Registro y validación de datos:

El sistema incluye verificaciones para garantizar que la información ingresada por el usuario sea válida:

Registro de huéspedes:

```python
def guardar_huesped(nombre, apellido, documento, correo, telefono):
    archivo_nuevo = not os.path.exists("huespedes.csv")
    with open("huespedes.csv", mode="a", newline="") as archivo:
        escritor = csv.writer(archivo)
        if archivo_nuevo:
            escritor.writerow(["Nombre", "Apellido", "Documento", "Correo", "Telefono"])
        escritor.writerow([nombre, apellido, documento, correo, telefono])
```

Primero se deben registrar los datos de los clientes y se guardarán en el archivo huéspedes.csv, que se creará automáticamente. El sistema se actualiza cuando se registra un huésped nuevo, de esta forma se facilita la gestión posterior de las reservas y los reportes.

```python
def registrar_huesped():
    nombre = input("Ingrese el nombre del huésped: ")
    apellido = input("Ingrese el apellido del huésped: ")
    documento = input("Ingrese el documento del huésped: ")
    correo = input("Ingrese el correo electrónico del huésped: ")
    telefono = input("Ingrese el teléfono del huésped: ")

    errores = []

    if not validar_nombre(nombre):
        errores.append("Nombre inválido: mínimo 3 letras y solo letras.")
    if not validar_apellido(apellido):
        errores.append("Apellido inválido: mínimo 3 letras y solo letras.")
    if not validar_documento(documento):
        errores.append("Documento inválido: entre 3 y 15 dígitos.")
    if not validar_correo(correo):
        errores.append("Correo inválido.")
    if not validar_telefono(telefono):
        errores.append("Teléfono inválido: entre 7 y 15 dígitos.")

    if errores:
        print("\nSe encontraron errores:")
        for e in errores:
            print("-", e)
    else:
        guardar_huesped(nombre, apellido, documento, correo, telefono)
        print("\n Registro exitoso.")
```

Esta función se encarga de solicitar y validar los datos de un huésped antes de registrarlo. Es una de las partes más importantes del sistema, ya que garantiza que la información guardada en el sistema sea correcta y ordenada bajo validaciones como:

- Nombres y apellidos: deben tener al menos 3 caracteres, sin cifras ni símbolos.
- Documento de identidad: debe contener entre 3 y 15 dígitos numéricos.
- Correo electrónico: debe poseer un símbolo "@" y una extensión válida (por ejemplo: .com, .edu, etc.).

De esta manera se evitan datos mal escritos o incompletos, haciendo que las búsquedas o reservas sean más confiables.

Reserva del hotel:

```python
def huesped_registrado(documento):
    try:
        df = pd.read_csv("huespedes.csv")
        return documento in df["Documento"].astype(str).values
    except FileNotFoundError:
        return False

def obtener_habitaciones_disponibles(tipo):
    try:
        df = pd.read_csv("reservas.csv")
        habitaciones_ocupadas = df["NumeroHabitacion"].astype(int).tolist()
    except (FileNotFoundError, KeyError):
        habitaciones_ocupadas = []

    if tipo == "estandar":
        todas = list(range(1, 31))
    elif tipo == "suite":
        todas = list(range(31, 41))
    else:
        return []

    disponibles = [h for h in todas if h not in habitaciones_ocupadas]
    return disponibles
```

Con la primera función se asegura que el documento del huésped esté previamente registrado y en el archivo. Luego, lee las habitaciones libres según el tipo de habitación elegida, y crea la lista correspondiente de las que están desocupadas.

```python
def realizar_reserva():
    print("\n REALIZAR RESERVA ")
    documento = input("Ingrese el documento del huésped: ")

    if not huesped_registrado(documento):
        print(" El huésped no está registrado.")
        return

    tipo = input("Tipo de habitación (estandar / suite): ").lower()
    if tipo not in ["estandar", "suite"]:
        print(" Tipo de habitación no válido.")
        return

    disponibles = obtener_habitaciones_disponibles(tipo)
    if not disponibles:
        print(f" No hay habitaciones {tipo} disponibles.")
        return

    print("Habitaciones disponibles:", disponibles)
    numero_hab = disponibles[0]

    fecha_ingreso_str = input("Ingrese la fecha de ingreso (AAAA-MM-DD): ")
    try:
        fecha_ingreso = datetime.strptime(fecha_ingreso_str, "%Y-%m-%d")
    except ValueError:
        print(" Formato de fecha inválido.")
        return

    try:
        noches = int(input("Ingrese el número de noches: "))
        if noches <= 0:
            raise ValueError
    except ValueError:
        print(" Número de noches inválido.")
        return
```

Primero verifica si el huésped está registrado, luego las habitaciones disponibles, y pide al usuario la fecha del ingreso y el número de noches del huésped, y calcula la fecha de salida automáticamente.

```
fecha_salida = fecha_ingreso + timedelta(days=noches)
costo_noche = 120000 if tipo == "estandar" else 250000
total = costo_noche * noches

print("\n RESERVA CONFIRMADA")
print(f"Huésped: {documento}")
print(f"Tipo: {tipo}")
print(f"Habitación: {numero_hab}")
print(f"Ingreso: {fecha_ingreso.date()} - Salida: {fecha_salida.date()}")
print(f"Noches: {noches}")
print(f"Total: ${total:,}")

archivo_nuevo = not os.path.exists("reservas.csv")
with open("reservas.csv", mode="a", newline="") as archivo:
    writer = csv.writer(archivo)
    if archivo_nuevo:
        writer.writerow(["Documento", "Tipo", "NumeroHabitacion", "FechaIngreso", "FechaSalida", "Noches", "Total"])
    writer.writerow([documento, tipo, numero_hab, fecha_ingreso.date(), fecha_salida.date(), noches, total])
```

Aquí, se calcula el total a pagar dependiendo del costo de la habitación (suite=250.000 y estándar= 120.000), luego, muestra el comprobante de la reserva que arroja el nombre, documento, el tipo y número de habitación, la fecha de ingreso y salida, el número de noches y el total a pagar.
Seguidamente abre el archivo de las reservas y la guarda.

Check out:

```
[13] #realizar la check-out

     def registrar_salida():
         print("\n REGISTRAR SALIDA ")
         documento = input("Ingrese el documento del huésped: ")

         try:
             df_reservas = pd.read_csv("reservas.csv")
             df_huespedes = pd.read_csv("huespedes.csv")
         except FileNotFoundError:
             print(" No hay registros para buscar.")
             return

         # Buscar reserva por documento
         reservas = df_reservas[df_reservas["Documento"].astype(str) == documento]

         if reservas.empty:
             print(" No se encontró ninguna reserva activa para ese documento.")
             return

         # Tomar la primera reserva encontrada (suponemos una sola)
         reserva = reservas.iloc[0]
         tipo = reserva["Tipo"]
         habitacion = reserva["NumeroHabitacion"]
         fecha_ingreso = reserva["FechaIngreso"]
         fecha_salida = reserva["FechaSalida"]
         noches = reserva["Noches"]
         total = reserva["Total"]
```

Para poder realizar el proceso de salida (check out) se verifica que el huésped tenga una reserva activa en los archivos reservas.csv usando el documento del cliente, si se encuentra la reserva el sistema carga los datos (tipo de habitación, número de habitación, fechas, cantidad de noches en las que se hospedó y total a pagar)

```python
# Buscar nombre del huésped en huespedes.csv
huesped = df_huespedes[df_huespedes["Documento"].astype(str) == documento]
if huesped.empty:
    print(" Huésped no encontrado en archivo de huéspedes.")
    nombre = "Desconocido"
else:
    nombre = huesped.iloc[0]["Nombre"] + " " + huesped.iloc[0]["Apellido"]

# factura
print("\n----- FACTURA DE HOSPEDAJE -----")
print(f"Huésped: {nombre}")
print(f"Documento: {documento}")
print(f"Tipo de habitación: {tipo}")
print(f"Número de habitación: {habitacion}")
print(f"Fecha de ingreso: {fecha_ingreso}")
print(f"Fecha de salida: {fecha_salida}")
print(f"Noches: {noches}")
print(f"Total a pagar: ${int(total):,}")

confirmar = input("\n¿Confirmar salida y liberar habitación? (si/no): ").lower()
if confirmar == "si":
    # Eliminar la reserva de reservas.csv
    df_reservas = df_reservas[df_reservas["Documento"].astype(str) != documento]
    df_reservas.to_csv("reservas.csv", index=False)
    print(" Reserva eliminada. Habitación liberada.")
else:
    print("Operación cancelada.")
```

Posteriormente a la generación detallada de la factura, se solicita información para realizar la salida y liberar la habitación, si el usuario acepta, se elimina la reserva. Así, podremos tener una información actualizada del estado de las habitaciones para futuros nuevos registros.

Funciones del menú del administrador:

se crea un menú interno al que solo tendrá acceso el administrador.

```
def mostrar_huespedes():
    try:
        df = pd.read_csv("huespedes.csv")
        print("\n--- HUÉSPEDES REGISTRADOS ---")
        print(df.to_string(index=False))
    except FileNotFoundError:
        print(" No hay huéspedes registrados aún.")

def mostrar_reservas():
    try:
        df = pd.read_csv("reservas.csv")
        print("\n--- RESERVAS ACTIVAS ---")
        print(df.to_string(index=False))
    except FileNotFoundError:
        print(" No hay reservas registradas aún.")

def buscar_reserva_por_documento():
    documento = input("Ingrese el documento a buscar: ")
    try:
        df = pd.read_csv("reservas.csv")
        resultado = df[df["Documento"].astype(str) == documento]
        if resultado.empty:
            print(" No se encontró ninguna reserva con ese documento.")
        else:
            print("\n--- RESERVAS ENCONTRADAS ---")
            print(resultado.to_string(index=False))
    except FileNotFoundError:
        print(" No hay reservas registradas aún.")
```

La primera función lee el archivo de los huéspedes registrados y los muestra en forma de tabla, y es la primera opción del menú del administrador: 1. ver huéspedes registrados.

La segunda función lee el archivo de las reservas y muestra las que están activas actualmente, y es la segunda opción del menú del administrador: 2. ver reservas activas.

La tercera función pide el número de documento del huésped, busca en el archivo de las reservas y muestra la reserva del documento solicitado, es la tercera opción del menú del administrador: 3. buscar reserva por documento.

Reportes de administrador:

```python
def reportes_administrador():
    print("\n REPORTES \n")
    try:
        df_huespedes = pd.read_csv("huespedes.csv")
        df_reservas = pd.read_csv("reservas.csv")
    except FileNotFoundError:
        print(" Archivos de datos no encontrados.")
        return

    # 1. Total de huéspedes registrados
    total_huespedes = len(df_huespedes)

    # 2. Total de habitaciones ocupadas actualmente
    habitaciones_ocupadas = df_reservas["NumeroHabitacion"].nunique()

    # 3. Total de habitaciones disponibles (hay 40 en total: 1-30 estándar, 31-40 suite)
    total_habitaciones = 40
    habitaciones_disponibles = total_habitaciones - habitaciones_ocupadas

    # 4. Total de ingresos generados
    ingresos_totales = df_reservas["Total"].sum()

    # 5. Tiempo promedio de estancia por huésped (en noches)
    promedio_noches = df_reservas.groupby("Documento")["Noches"].sum().mean()

    # 6. Lista de huéspedes con historial de reservas
    documentos_con_reserva = df_reservas["Documento"].unique().astype(str)
    historial = df_huespedes[df_huespedes["Documento"].astype(str).isin(documentos_con_reserva)]
```

```python
    # 7. Huésped con más noches hospedadas
    suma_noches = df_reservas.groupby("Documento")["Noches"].sum()
    doc_mas_noches = suma_noches.idxmax()
    max_noches = suma_noches.max()
    nombre_max = df_huespedes[df_huespedes["Documento"].astype(str) == str(doc_mas_noches)][["Nombre", "Apellido"]].iloc[0].to_list()
    nombre_max = " ".join(nombre_max)

    # 8. Huésped con menos noches hospedadas
    doc_menos_noches = suma_noches.idxmin()
    min_noches = suma_noches.min()
    nombre_min = df_huespedes[df_huespedes["Documento"].astype(str) == str(doc_menos_noches)][["Nombre", "Apellido"]].iloc[0].to_list()
    nombre_min = " ".join(nombre_min)

    # Mostrar todos los reportes
    print(f" Total de huéspedes registrados: {total_huespedes}")
    print(f" Total de habitaciones ocupadas: {habitaciones_ocupadas}")
    print(f" Total de habitaciones disponibles: {habitaciones_disponibles}")
    print(f" Total de ingresos generados: ${ingresos_totales:,}")
    print(f" Tiempo promedio de estancia por huésped: {promedio_noches:.2f} noches")
    print("\n Lista de huéspedes con historial de reservas:")
    print(historial[["Nombre", "Apellido", "Documento"]].to_string(index=False))

    print(f"\n Huésped con más noches: {nombre_max} ({max_noches} noches)")
    print(f" Huésped con menos noches: {nombre_min} ({min_noches} noches)")
```

Esta función genera reportes estadísticos para el administrador del hotel. Muestra el total de huéspedes registrados, las habitaciones ocupadas y disponibles, ingresos totales, promedio de noches por huésped, la lista de huéspedes que han hecho reservas, cuales son los tienen más noches y los que tienen menos noches.

Para ello, se agrupan las reservas por documento y se suma la cantidad de noches hospedadas.
Luego se busca el nombre completo de cada persona, y se muestra junto con su total de noches. Esta información se obtiene de los archivos csv usando operaciones de filtrado, agrupación y agregación con la librería pandas

Visualización del menú del administrador:

```
def menu_administrador():
    clave = input(" Ingrese la contraseña del administrador: ")
    if clave != "hotel890":
        print(" Contraseña incorrecta. Acceso denegado.")
        return

    while True:
        print("\n MENÚ DEL ADMINISTRADOR ")
        print("1. Ver huéspedes registrados")
        print("2. Ver reservas activas")
        print("3. Buscar reserva por documento")
        print("4. Ver reportes generales")
        print("5. Volver al menú principal")

        opcion = input("Seleccione una opción (1-5): ")

        if opcion == "1":
            mostrar_huespedes()
        elif opcion == "2":
            mostrar_reservas()
        elif opcion == "3":
            buscar_reserva_por_documento()
        elif opcion == "4":
            reportes_administrador()
        elif opcion == "5":
            break
        else:
            print(" Opción inválida.")
```

Se crea una clave a la que solo tiene acceso el administrador, en este caso la clave es “hotel890” y así da ingreso al menú que le mostrará la información seleccionada por él. Se muestran dentro del bucle del **while** todas las opciones del menú, hasta la opción 5 que lo lleva al menú principal del sistema, rompiendo así el bucle. además, invalida si se ingresa una opción que no esté en el menú.

Menú principal:

```
def menu():
    while True:
        print("\n HOTEL BUENDESCANSO ")
        print("1. Registrar huésped")
        print("2. Realizar reserva")
        print("3. Registrar salida (check-out)")
        print("4. Administrador")
        print("5. Salir")

        opcion = input("Seleccione una opción (1-5): ")

        if opcion == "1":
            registrar_huesped()
        elif opcion == "2":
            realizar_reserva()
        elif opcion == "3":
            registrar_salida()
        elif opcion == "4":
            menu_administrador()
        elif opcion == "5":
            print(" ¡sistema hotel buendescanso!")
            break
        else:
            print(" Opción inválida. Intente de nuevo.")
```

Se crea el menú principal con las opciones correspondientes a lo que el sistema está destinado a hacer, que es el registro del huésped, la reserva, registrar la salida y la opción privada del administrador, también se le agrega la opción de salir para cerrar el sistema del hotel.

Visualización de Reportes Gráficos con Matplotlib:

Durante el desarrollo del sistema, se implementaron 8 gráficos diferentes que permiten interpretar de forma rápida y clara la información registrada en los archivos del sistema (como reservas, ocupación e ingresos).

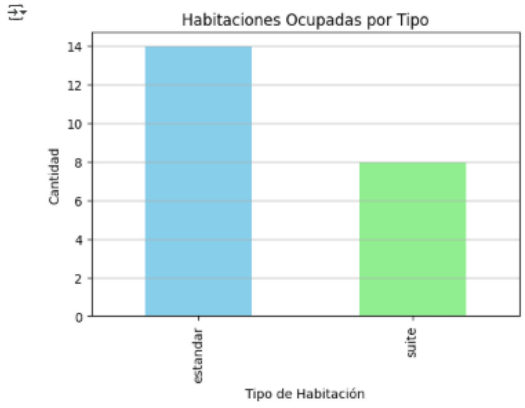
1. **Gráfica de barras:**

Se usa para conocer las habitaciones ocupadas por tipo (estándar/suite) leyendo la información del archivo csv de reservas, en el gráfico lo podemos observar de la siguiente manera:

- Eje horizontal: tipos de habitación
- Eje vertical: Cantidad de veces que se ha ocupado cada tipo

```
[43] def grafica_barras_tipo_habitacion():
        try:
            df = pd.read_csv("reservas.csv")
            conteo = df["Tipo"].value_counts()
            plt.figure(figsize=(6,4))
            conteo.plot(kind="bar", color=["skyblue", "lightgreen"])
            plt.title("Habitaciones Ocupadas por Tipo")
            plt.xlabel("Tipo de Habitación")
            plt.ylabel("Cantidad")
            plt.grid(axis="y")
            plt.show()
        except:
            print("Error al generar la gráfica.")
```

```
[44] grafica_barras_tipo_habitacion()
```



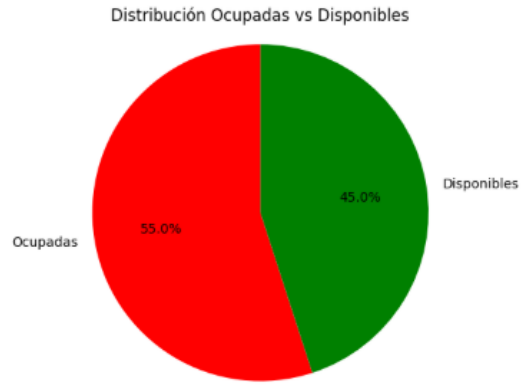
2. **Gráfica circular (pie chart):**

Esta herramienta permite visualizar de manera rápida qué porcentaje de las habitaciones del hotel se encuentran ocupadas y cuáles están disponibles usando los datos del archivo csv de reservas.

- La sección roja representa las habitaciones ocupadas.
- La sección verde representa las habitaciones disponibles.
- Se indica el porcentaje exacto de cada categoría (55% ocupadas y 45% disponibles).

```python
def grafica_pie_ocupadas_vs_disponibles():
    try:
        df = pd.read_csv("reservas.csv")
        ocupadas = df["NumeroHabitacion"].nunique()
        total = 40
        disponibles = total - ocupadas
        plt.figure(figsize=(5,5))
        plt.pie([ocupadas, disponibles], labels=["Ocupadas", "Disponibles"],
                colors=["red", "green"], autopct="%1.1f%%", startangle=90)
        plt.title("Distribución Ocupadas vs Disponibles")
        plt.axis("equal")
        plt.show()
    except:
        print("Error al generar la gráfica.")
```

```python
grafica_pie_ocupadas_vs_disponibles()
```



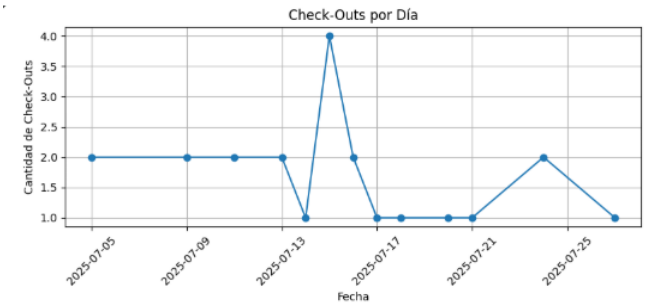
3. Gráfica de línea:

Esta gráfica nos permite visualizar los check out por día, es decir, muestra cuántos huéspedes han abandonado el hotel en cada fecha específica. Nos permite identificar los días con mayor rotación de habitaciones.

- Eje horizontal: Fechas de salida.
- Eje vertical: Cantidad de personas que se fueron en esa fecha.

```
[47] def grafica_lineas_checkouts_por_dia():
        try:
            df = pd.read_csv("reservas.csv")
            fechas = pd.to_datetime(df["FechaSalida"])
            conteo = fechas.value_counts().sort_index()
            plt.figure(figsize=(8,4))
            plt.plot(conteo.index, conteo.values, marker='o')
            plt.title("Check-Outs por Día")
            plt.xlabel("Fecha")
            plt.ylabel("Cantidad de Check-Outs")
            plt.grid(True)
            plt.xticks(rotation=45)
            plt.tight_layout()
            plt.show()
        except:
            print("Error al generar la gráfica.")
```

```
] grafica_lineas_checkouts_por_dia()
```



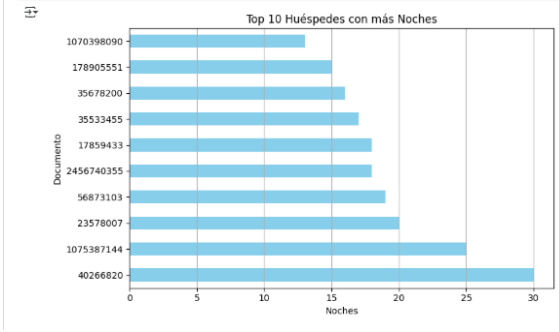
4. **Gráfica de barras horizontal:**

Este gráfico muestra los 10 huéspedes que han pasado más noches en el hotel, ordenados de mayor a menor. Sirve para identificar a los clientes frecuentes y visualizar quiénes han tenido estancias más largas.

- Eje horizontal: Cantidad de noches
- Eje vertical: Documento de identificación del huésped

```
[49] def grafica_barras_horizontal_noches_top10():
        try:
            df = pd.read_csv("reservas.csv")
            top10 = df.groupby("Documento")["Noches"].sum().sort_values(ascending=False).head(10)
            plt.figure(figsize=(8,5))
            top10.plot(kind="barh", color="skyblue")
            plt.title("Top 10 Huéspedes con más Noches")
            plt.xlabel("Noches")
            plt.ylabel("Documento")
            plt.grid(axis="x")
            plt.tight_layout()
            plt.show()
        except:
            print("Error al generar la gráfica.")
```

```
[50] grafica_barras_horizontal_noches_top10()
```



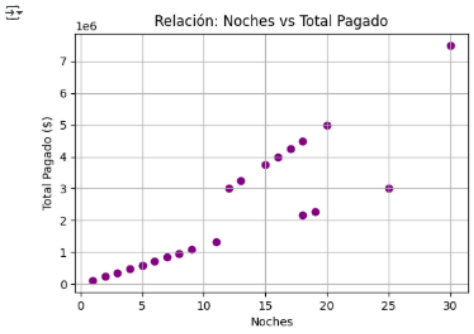
5. Gráfica de dispersión (scatter plot) :

El gráfico de dispersión muestra las tarifas y estancias por cada huésped. Cada punto del gráfico representa una reserva individual, mostrando cuánto pagó un cliente según el tiempo que permaneció en el hotel.

- El Eje horizontal muestra la cantidad de noches reservadas.
- El Eje vertical muestra el total pagado en dinero por esa reserva.

```
[51] def grafica_dispersion_noches_vs_total():
        try:
            df = pd.read_csv("reservas.csv")
            resumen = df.groupby("Documento")[["Noches", "Total"]].sum()
            plt.figure(figsize=(6,4))
            plt.scatter(resumen["Noches"], resumen["Total"], color="purple")
            plt.title("Relación: Noches vs Total Pagado")
            plt.xlabel("Noches")
            plt.ylabel("Total Pagado ($)")
            plt.grid(True)
            plt.show()
        except:
            print("Error al generar la gráfica.")
```

```
[52] grafica_dispersion_noches_vs_total()
```

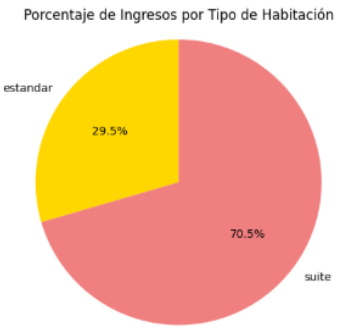


6. Gráfica de pastel:

El gráfico de pastel muestra la proporción de ingresos que genera cada tipo de habitación (estándar y suite). Se calculan los totales recibidos por cada tipo, y se representan en forma gráfica para así poder ver la cantidad de ganancias que genera cada tipo de habitación. (29,5% estándar y 70,5% suite)

```
[53] def grafica_pastel_ingresos_por_tipo():
         try:
             df = pd.read_csv("reservas.csv")
             ingresos = df.groupby("Tipo")["Total"].sum()
             plt.figure(figsize=(5,5))
             plt.pie(ingresos, labels=ingresos.index, autopct="%1.1f%%", startangle=90,
                     colors=["gold", "lightcoral"])
             plt.title("Porcentaje de Ingresos por Tipo de Habitación")
             plt.axis("equal")
             plt.show()
         except:
             print("Error al generar la gráfica.")
```

```
[54] grafica_pastel_ingresos_por_tipo()
```

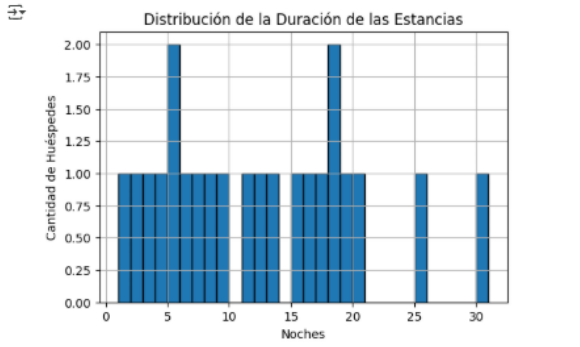


7. **Histograma:**

   Se utiliza para visualizar cuántos huéspedes se quedaron determinado número de noches en el hotel. Cada barra representa una duración específica (1 noche, 2 noches, 3 noches, etc), y su altura indica cuántas veces se repitió esa duración.

```
[55] def grafica_histograma_noches():
        try:
            df = pd.read_csv("reservas.csv")
            noches = df["Noches"]
            plt.figure(figsize=(6,4))
            plt.hist(noches, bins=range(1, noches.max()+2), edgecolor="black")
            plt.title("Distribución de la Duración de las Estancias")
            plt.xlabel("Noches")
            plt.ylabel("Cantidad de Huéspedes")
            plt.grid(True)
            plt.show()
        except:
            print("Error al generar la gráfica.")
```

```
[56] grafica_histograma_noches()
```



8. **Gráfica combinada (barras + líneas):**

Permite observar simultáneamente dos aspectos clave del hotel: los ingresos diarios (mostrados con barras) y la cantidad de huéspedes ingresados por día (mostrados con una línea).

El sistema agrupa los datos por fecha de ingreso y calcula tanto el total pagado como la cantidad de reservas hechas ese día.

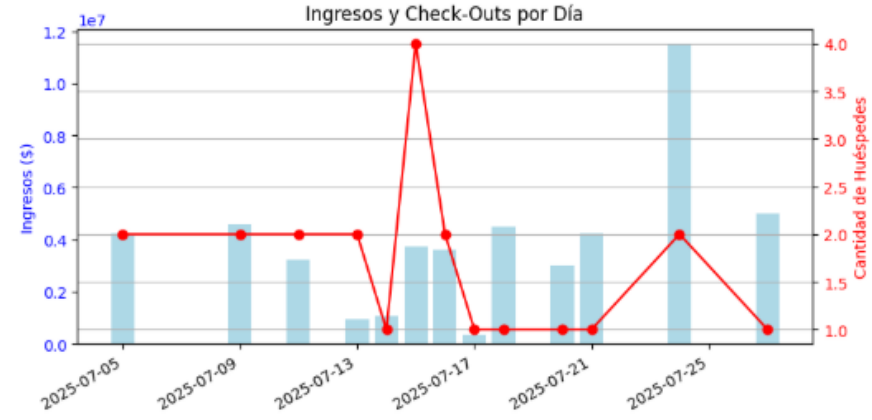
```
[57] def grafica_combinada_ingresos_y_checkouts():
        try:
            df = pd.read_csv("reservas.csv")
            df["FechaSalida"] = pd.to_datetime(df["FechaSalida"])
            resumen = df.groupby("FechaSalida").agg({"Total": "sum", "Documento": "count"}).rename(columns={"Documento": "Checkouts"})
            fig, ax1 = plt.subplots(figsize=(8,4))

            ax1.bar(resumen.index, resumen["Total"], color="lightblue", label="Ingresos diarios")
            ax1.set_ylabel("Ingresos ($)", color="blue")
            ax1.tick_params(axis="y", labelcolor="blue")

            ax2 = ax1.twinx()
            ax2.plot(resumen.index, resumen["Checkouts"], color="red", marker="o", label="Check-Outs")
            ax2.set_ylabel("Cantidad de Huéspedes", color="red")
            ax2.tick_params(axis="y", labelcolor="red")

            plt.title("Ingresos y Check-Outs por Día")
            fig.autofmt_xdate()
            plt.grid(True)
            plt.tight_layout()
            plt.show()
        except:
            print("Error al generar la gráfica combinada.")
```

```
grafica_combinada_ingresos_y_checkouts()
```



Conclusión

El sistema Hotel Luna Azul cumple con los objetivos propuestos, permitiendo gestionar de forma básica la operación diaria de un hotel, usando programación estructurada en Python y funcionalidades de lectura, escritura de archivos y visualización de datos. Se incorporaron buenas prácticas de validación de entradas, organización modular del código, y una interfaz de usuario basada en menú por consola.

Gracias al uso de librerías estándar, el sistema es ligero, portable y puede seguir ampliándose con más características como historial de reservas por huésped, facturación, o interfaz gráfica.