Análisis de Datos de Clientes de un Hotel

Este análisis tiene como objetivo explorar los datos de clientes de un hotel, identificar patrones de comportamiento y evaluar factores clave como la satisfacción, la duración de la estancia y el gasto de los huéspedes.

Utilizaremos técnicas de análisis exploratorio de datos (EDA), estadísticas descriptivas y modelos de Machine Learning para extraer información relevante.

Índice de Contenidos

In [5]:
```
import numpy as np
import os
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_absolute_percentage_error, r2_score
from sklearn.linear_model import LinearRegression, Lasso, Ridge, ElasticNet
```

1. Carga de Datos

Para iniciar el análisis, cargamos el dataset de clientes del hotel. Este dataset contiene información relevante sobre las características de los clientes, su estancia y su nivel de satisfacción.

Pasos realizados:

1. **Cargamos** los datos desde un archivo CSV.
2. **Eliminamos** columnas innecesarias para optimizar el análisis.
3. **Inspeccionamos** la estructura de los datos con `df.info()` para verificar su integridad.

In [6]:
```
path = "../data/"
df = pd.read_csv(os.path.join(path, "datosclienteshotel.csv"), sep=";")
df.drop(columns=["Unnamed: 0"], inplace=True)
print(f"Dataset cargado con {df.shape[0]} filas y {df.shape[1]} columnas.")
df.head()
```

```
Dataset cargado con 5000 filas y 9 columnas.
```

Out[6]:

	Nombre	Apellido	Sexo	Nacionalidad	Edad	Número Días de Estancia	Precio Pagado	Tipo Habitación	Puntuación satisfacción
0	Fernando	Martínez	hombre	Europea	55	15	844.80	doble	5.0
1	Carmen	López	mujer	No Europea	38	13	676.13	individual	7.0
2	Andrea	González	mujer	Europea	19	9	470.31	individual	8.7
3	Abel	González	hombre	Europea	40	11	624.10	individual	5.3
4	María	García	mujer	Europea	57	16	863.46	doble	5.8

2. Limpieza y Preparación de Datos

Antes de realizar el análisis, es fundamental asegurarnos de que los datos sean correctos y estén en un formato adecuado para su procesamiento.

Transformaciones aplicadas:

- **Conversión de columnas** a formato numérico para evitar problemas en cálculos posteriores.
- **Manejo de valores nulos** mediante estrategias específicas para cada variable.

Estas transformaciones nos permitirán obtener insights más precisos y evitar errores en los modelos estadísticos y de Machine Learning.

In [7]:
```
df['Edad'] = pd.to_numeric(df['Edad'], errors='coerce')
df['Número Días de Estancia'] = pd.to_numeric(df['Número Días de Estancia'], errors='coerce')
df['Precio Pagado'] = pd.to_numeric(df['Precio Pagado'], errors='coerce')
df['Puntuación satisfacción'] = pd.to_numeric(df['Puntuación satisfacción'], errors='coerce')
```
In [8]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Nombre                 5000 non-null   object 
 1   Apellido               5000 non-null   object 
 2   Sexo                   5000 non-null   object 
 3   Nacionalidad           5000 non-null   object 
 4   Edad                   5000 non-null   int64  
 5   Número Días de Estancia  5000 non-null   int64  
 6   Precio Pagado          5000 non-null   float64
 7   Tipo Habitación        5000 non-null   object 
 8   Puntuación satisfacción  5000 non-null   float64
dtypes: float64(2), int64(2), object(5)
memory usage: 351.7+ KB
```

3. Análisis Exploratorio de Datos (EDA)

El análisis exploratorio de datos nos ayuda a entender mejor la distribución y relaciones entre las variables.

Análisis Unidimensional:

- Distribución de variables categóricas y numéricas.
- Tablas de frecuencia y gráficos de barras para identificar tendencias.
- Histogramas y boxplots para análisis de dispersión.

Análisis Bidimensional:

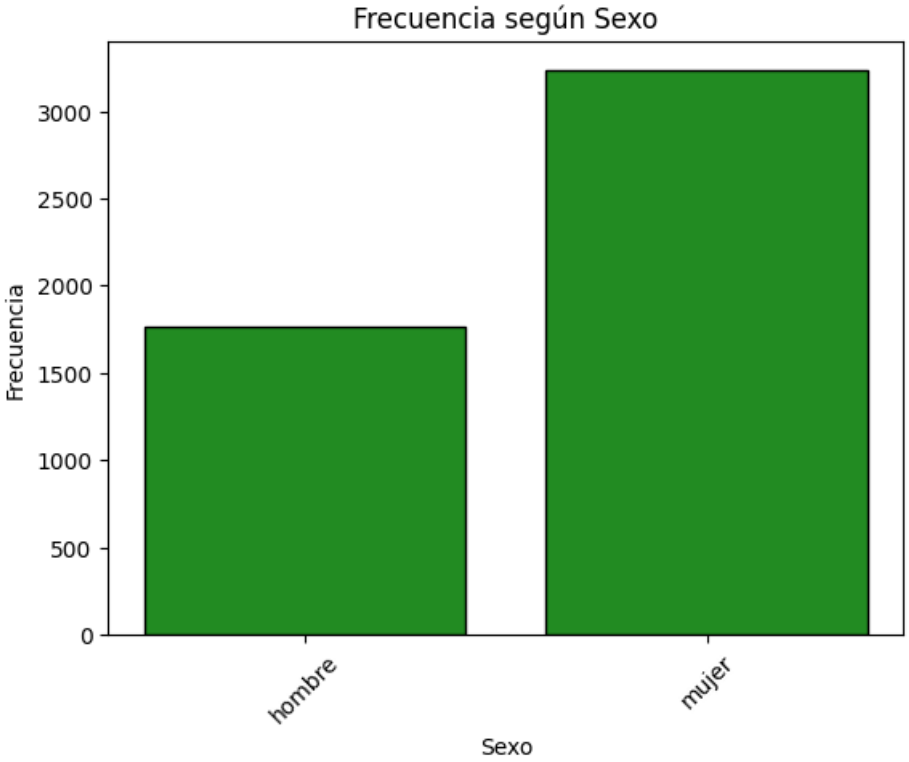
- Pairplots para explorar relaciones entre variables.
- Matriz de correlación para identificar asociaciones fuertes entre variables clave.

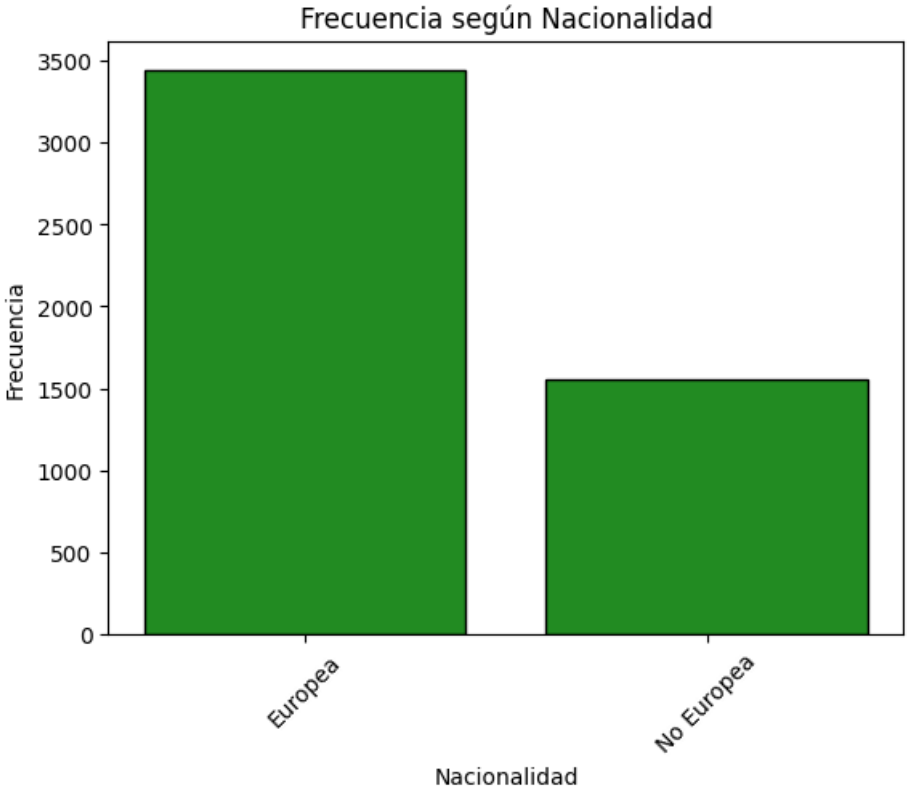
Resultados Clave:

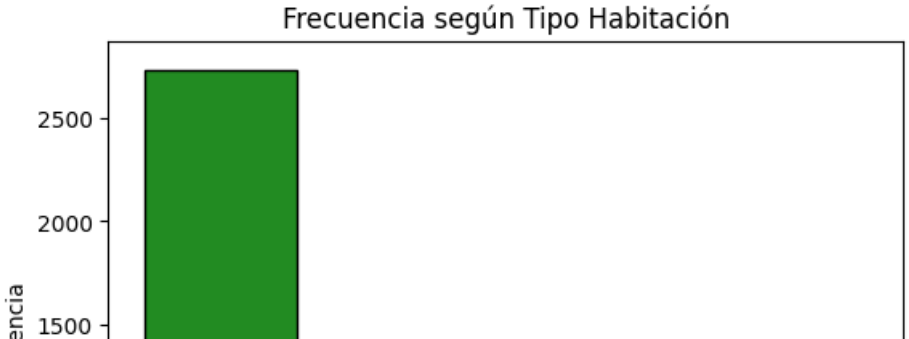
- Se observa una **fuerte correlación positiva** entre el número de días de estancia y el precio pagado.
- La **satisfacción de los clientes** tiende a disminuir a medida que aumenta la edad y el precio pagado.

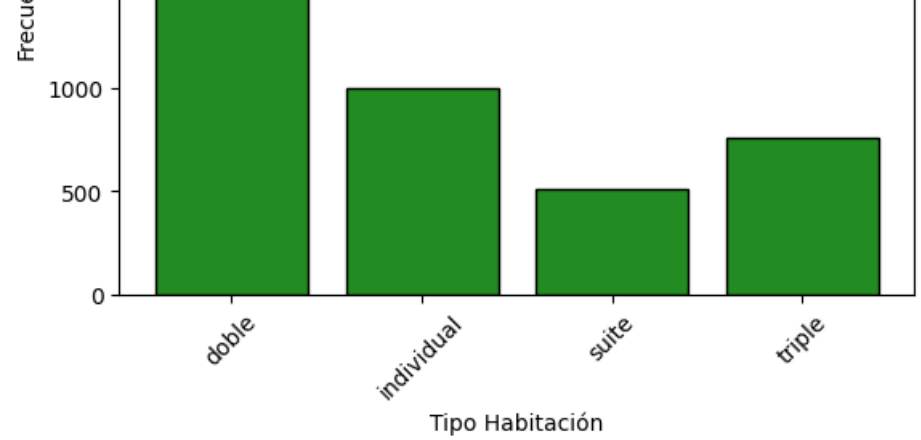
In [9]:
```
def generar_tabla_frecuencia(df, columna):
    df_freq = df.groupby(columna).agg(Freq=(columna, 'count')).reset_index()
    df_freq['Freq_Rel'] = 100 * df_freq['Freq'] / df_freq['Freq'].sum()
    df_freq['Freq_Acum'] = df_freq['Freq'].cumsum()
    df_freq['Freq_Rel_Acum'] = df_freq['Freq_Rel'].cumsum()
    return df_freq
```
In [10]:

```
# Análisis de variables categóricas
for var in ['Sexo', 'Nacionalidad', 'Tipo Habitación']:
    df_freq = generar_tabla_frecuencia(df, var)
    plt.bar(df_freq[var], df_freq['Freq'], color='forestgreen', edgecolor='black')
    plt.title(f'Frecuencia según {var}')
    plt.xlabel(var)
    plt.ylabel('Frecuencia')
    plt.xticks(rotation=45)
    plt.show()
```





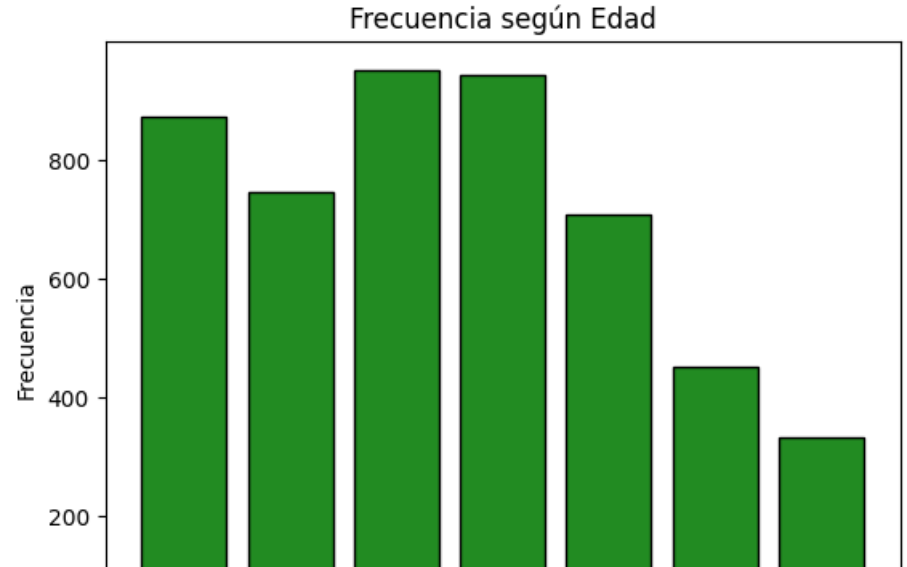


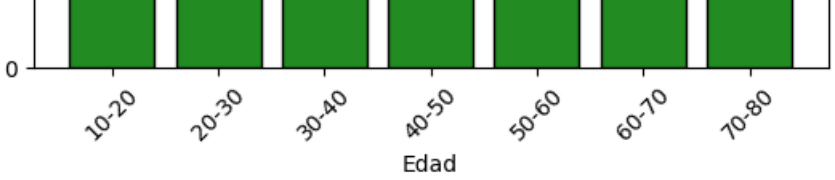


In [11]:

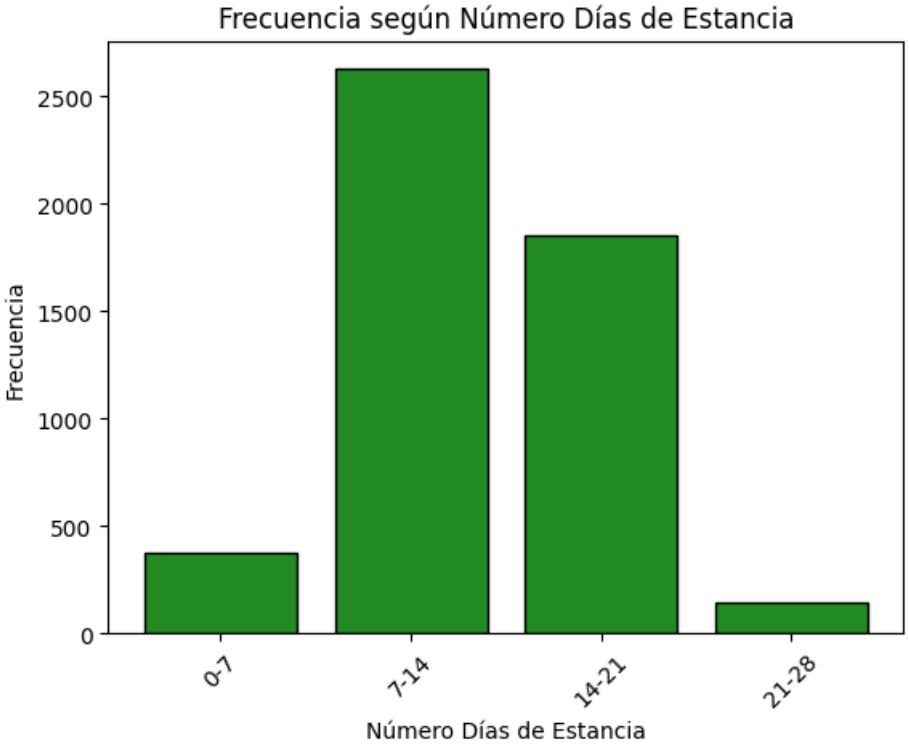
```
bins_dict = {
    'Edad': [10, 20, 30, 40, 50, 60, 70, 80],
    'Número Días de Estancia': [0, 7, 14, 21, 28],
    'Precio Pagado': [0, 300, 600, 900, 1200, 1500],
    'Puntuación satisfacción': [0, 2, 3, 4, 5, 6, 7, 8, 9, 10]
}

def agrupar_datos(df, columna, bins):
    df[columna] = pd.to_numeric(df[columna], errors='coerce')

    if df[columna].isna().all():
        print(f"⚠ La columna '{columna}' está vacía después de la conversión. No se puede agrupar.")
        return None

    labels = [f'{bins[i]}-{bins[i+1]}' for i in range(len(bins)-1)]

    df[f"{columna}_grupo"] = pd.cut(df[columna], bins=bins, labels=labels, right=True)

    return generar_tabla_frecuencia(df, f"{columna}_grupo")

for var, bins in bins_dict.items():
    df_freq = agrupar_datos(df, var, bins)

    if df_freq is not None:
        columna_agrupada = f"{var}_grupo"
        if columna_agrupada in df_freq.columns:
            plt.bar(df_freq[columna_agrupada], df_freq['Freq'], color='forestgreen', edgecolor='black')
            plt.title(f'Frecuencia según {var}')
            plt.xlabel(var)
            plt.ylabel('Frecuencia')
            plt.xticks(rotation=45)
            plt.show()
        else:
            print(f" La columna agrupada '{columna_agrupada}' no se encuentra en df_freq.")
    else:
        print(f"No se pudo generar la tabla de frecuencias para '{var}'.")
```

C:\Users\carlo\AppData\Local\Temp\ipykernel_23304\1350190192.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  df_freq = df.groupby(columna).agg(Freq=(columna, 'count')).reset_index()

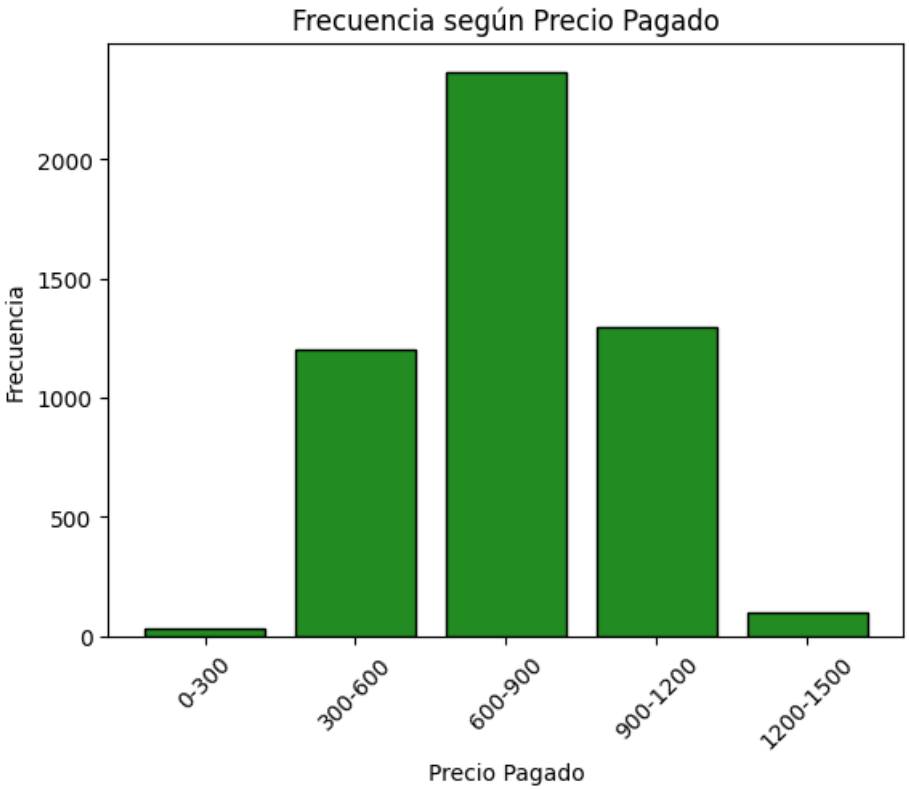




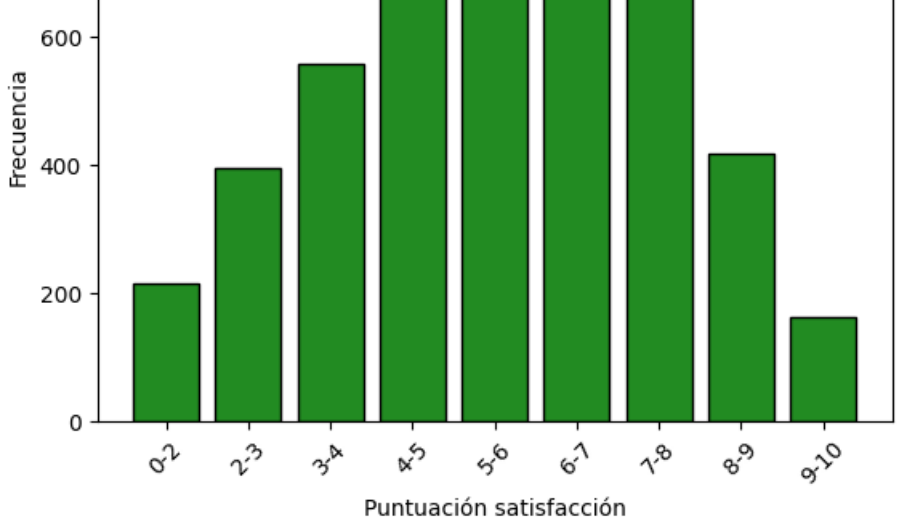
C:\Users\carlo\AppData\Local\Temp\ipykernel_23304\1350190192.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  df_freq = df.groupby(columna).agg(Freq=(columna, 'count')).reset_index()



C:\Users\carlo\AppData\Local\Temp\ipykernel_23304\1350190192.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  df_freq = df.groupby(columna).agg(Freq=(columna, 'count')).reset_index()



C:\Users\carlo\AppData\Local\Temp\ipykernel_23304\1350190192.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  df_freq = df.groupby(columna).agg(Freq=(columna, 'count')).reset_index()



In [12]:
```
print(df.dtypes)
```

```
Nombre                          object
Apellido                        object
Sexo                            object
Nacionalidad                    object
Edad                             int64
Número Días de Estancia          int64
Precio Pagado                  float64
Tipo Habitación                 object
Puntuación satisfacción        float64
Edad_grupo                    category
Número Días de Estancia_grupo    category
Precio Pagado_grupo              category
Puntuación satisfacción_grupo    category
dtype: object
```

In [13]:
```
df['Sexo'].describe()
```

Out[13]:
```
count      5000
unique        2
top       mujer
freq       3236
Name: Sexo, dtype: object
```

In [14]:
```
df['Nacionalidad'].describe()
```

Out[14]:
```
count        5000
unique          2
top       Europea
freq         3446
Name: Nacionalidad, dtype: object
```

In [15]:
```
df['Tipo Habitación'].describe()
```

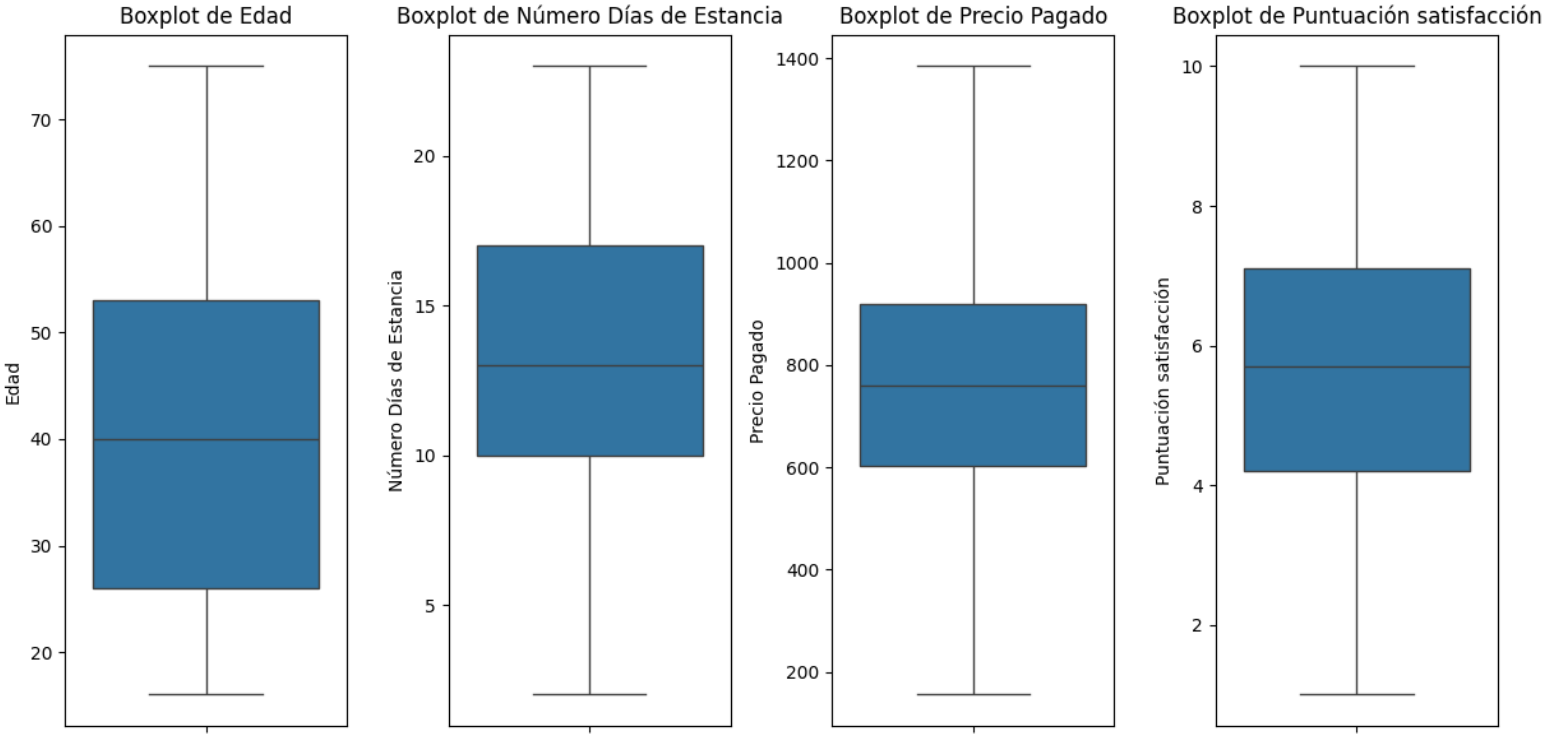
Out[15]:
```
count      5000
unique        4
top       doble
freq       2736
Name: Tipo Habitación, dtype: object
```

In [16]:
```
round(df['Número Días de Estancia'].describe(),4)
```

Out[16]:
```
count    5000.0000
mean       13.4946
std         4.1987
min         2.0000
25%        10.0000
50%        13.0000
75%        17.0000
max        23.0000
Name: Número Días de Estancia, dtype: float64
```

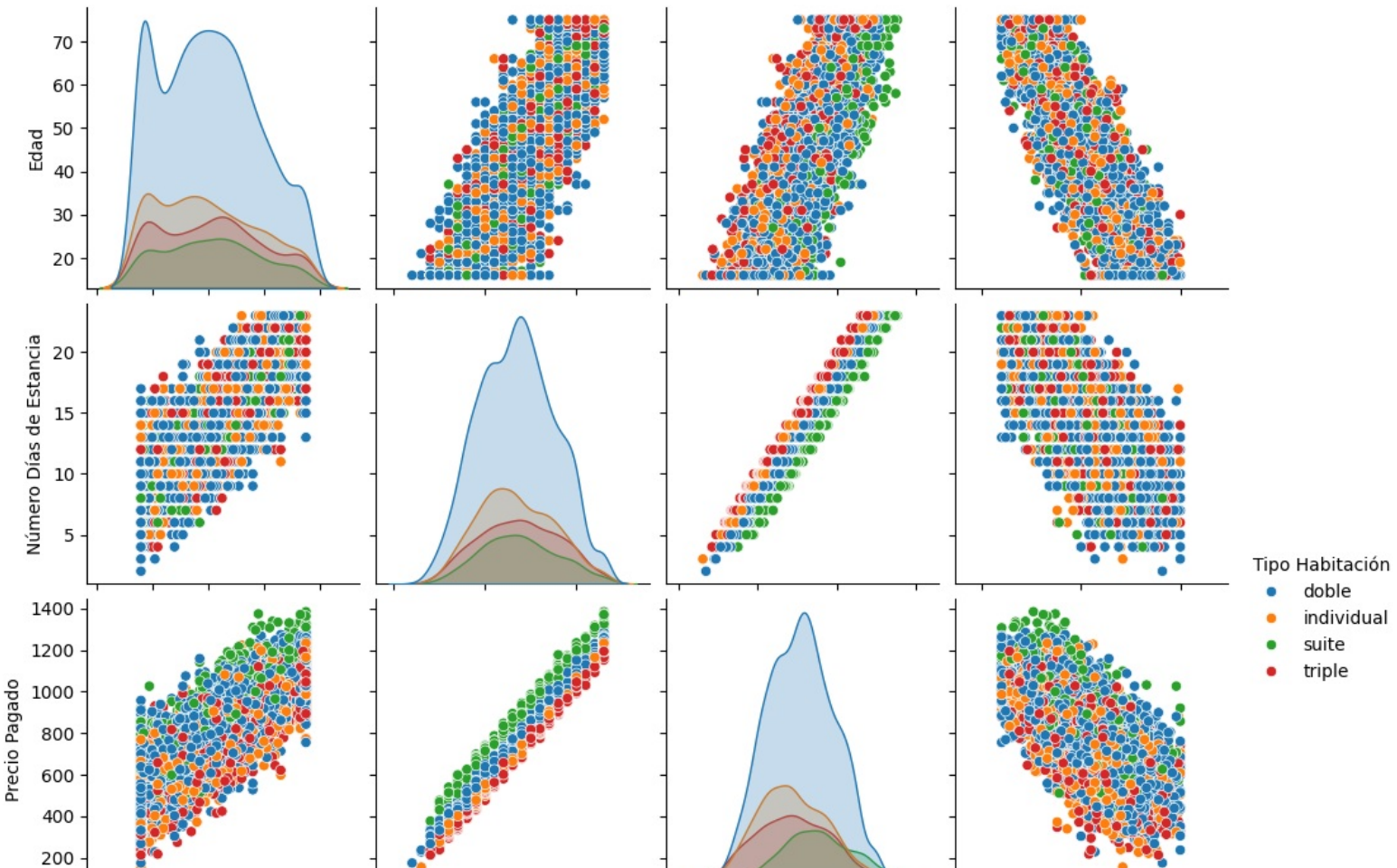
In [17]:

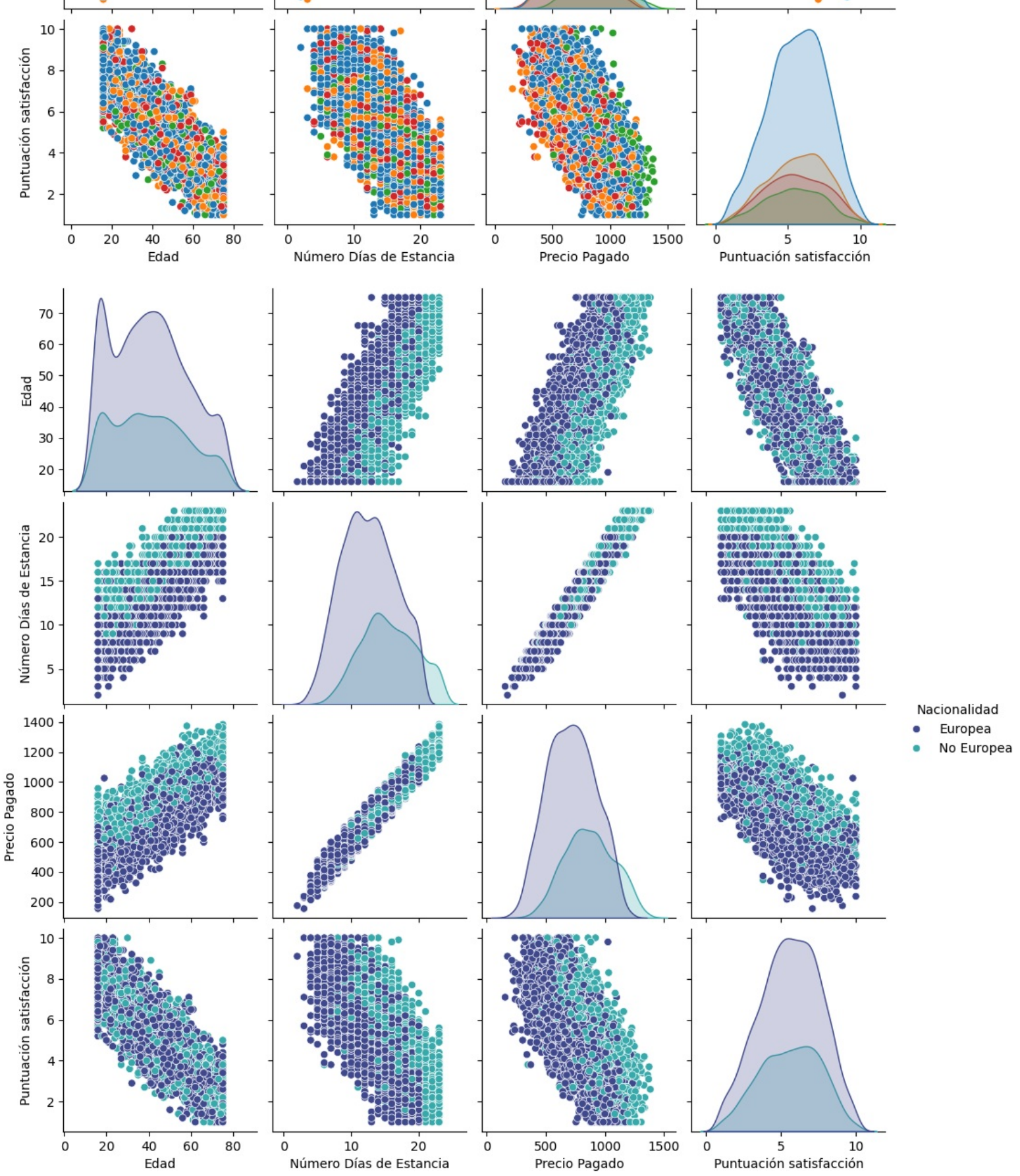
```
variables_numericas = ['Edad', 'Número Días de Estancia', 'Precio Pagado', 'Puntuación satisfacción']

plt.figure(figsize=(12, 6))
for i, col in enumerate(variables_numericas, 1):
    plt.subplot(1, len(variables_numericas), i)
    sns.boxplot(y=df[col])
    plt.title(f"Boxplot de {col}")

plt.tight_layout()
plt.show()
```



In [18]:

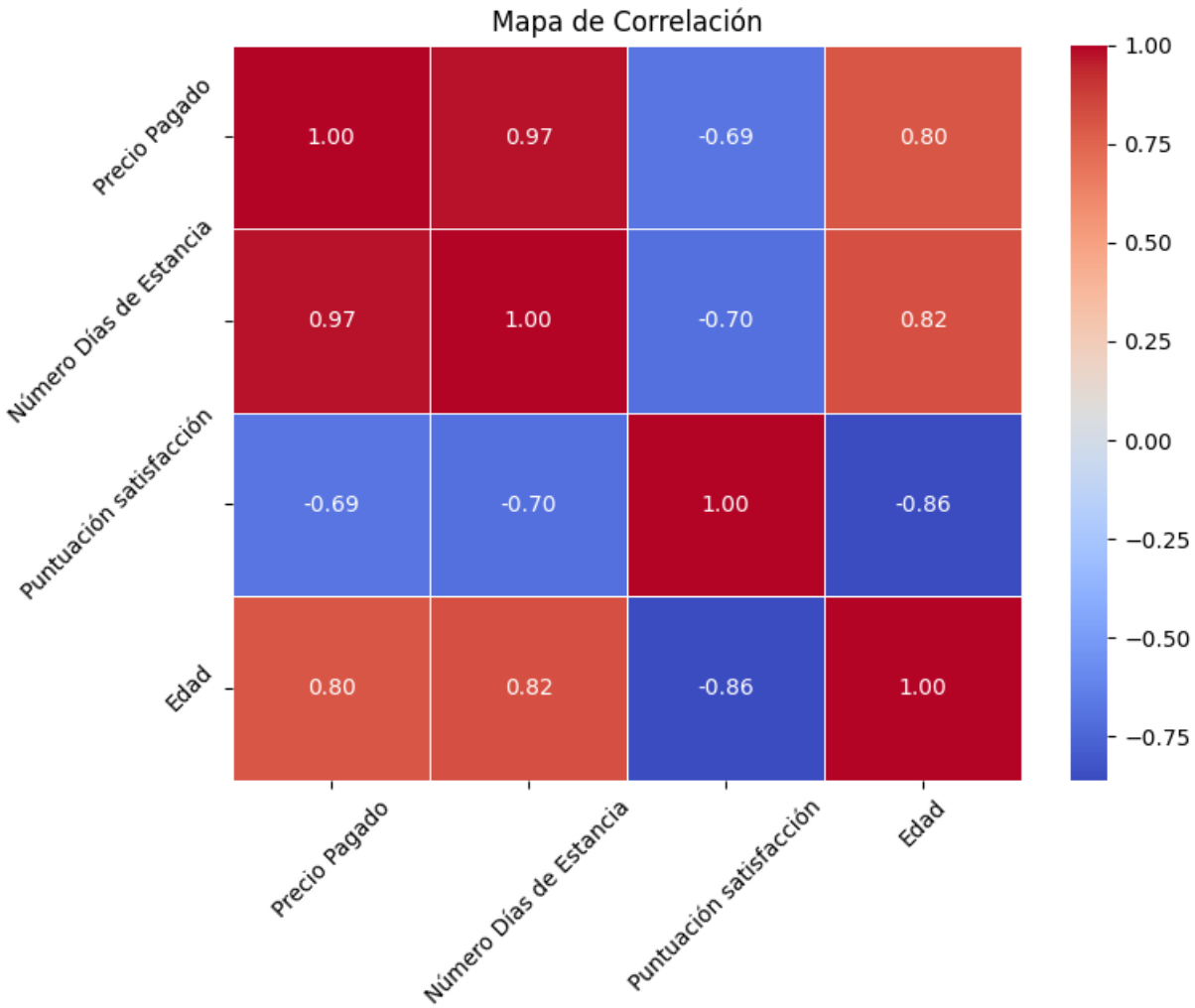
```
## Análisis Bidimensional
sns.pairplot(df, hue='Tipo Habitación', palette='tab10')
plt.show()
sns.pairplot(df, hue='Nacionalidad', palette='mako')
plt.show()
```





In [19]:

```
# Matriz de correlación
plt.figure(figsize=(8, 6))
sns.heatmap(df[['Precio Pagado', 'Número Días de Estancia', 'Puntuación satisfacción', 'Edad']].corr(), annot=True, cmap='coolwarm', fmt='
plt.title("Mapa de Correlación")
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.show()
```



4. Análisis Estadístico y Regresiones

Para profundizar en las relaciones entre variables, se aplicaron distintos modelos de regresión.

Regresión Lineal Simple:

- Relación entre la **edad** y la **duración de la estancia**.
- Evaluación de coeficientes y métricas de error (MAE, MSE, R^2).

Regresión Múltiple:

- Predicción del **precio pagado** en función de edad, satisfacción y días de estancia.
- Comparación entre valores reales y predichos.

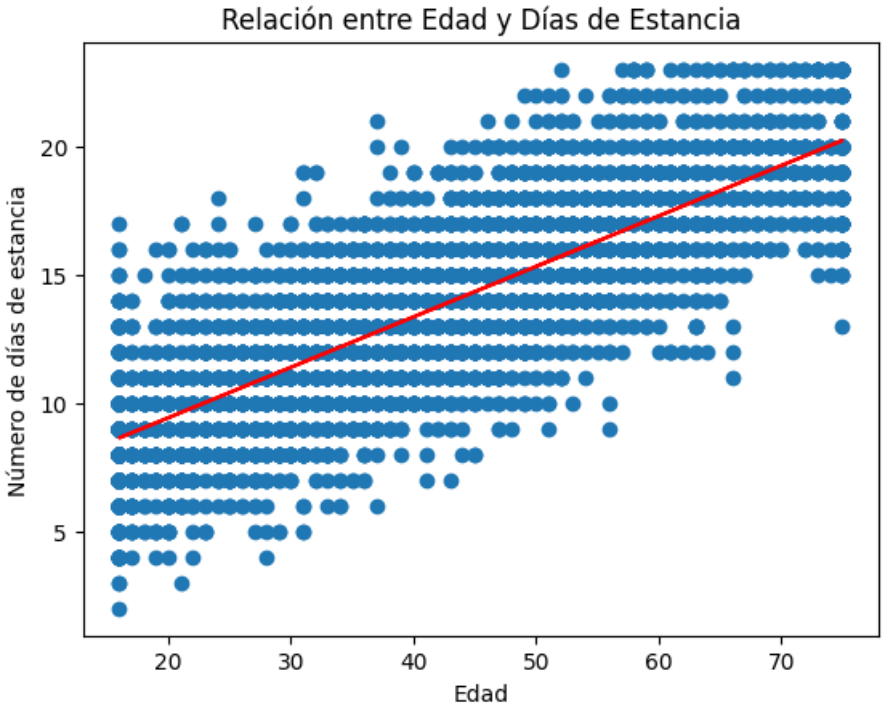
Resultados Clave:

- La estancia promedio de un cliente de 45 años se estima en **X días**.
- El modelo de regresión múltiple ofrece un coeficiente de determinación de **R^2=0.85**, indicando una buena capacidad predictiva.

Las regresiones nos permiten cuantificar el impacto de cada variable sobre la otra y realizar predicciones razonables sobre nuevos datos.

In [20]:
```
## Modelos de Regresión

def entrenar_modelo(x, y):
    model = LinearRegression()
    model.fit(x, y)
    y_pred = model.predict(x)
    return model, y_pred
```
In [21]:

```
# Regresión simple
x = df[["Edad"]]
y = df[["Número Días de Estancia"]]
modelo_edad_estancia, y_pred = entrenar_modelo(x, y)

plt.scatter(x, y)
plt.plot(x, y_pred, color='red')
plt.xlabel("Edad")
plt.ylabel("Número de días de estancia")
plt.title("Relación entre Edad y Días de Estancia")
plt.show()

print("Intercepto:", modelo_edad_estancia.intercept_)
print("Coeficiente:", modelo_edad_estancia.coef_)
print("R²:", r2_score(y, y_pred))
```




```
Intercepto: [5.50609546]
Coeficiente: [[0.19637715]]
R²: 0.6702162138109199
```

Nuestro modelo tiene la siguiente forma:**y = 0.1964x + 5.5061**

Tenemos un MAPE del 17% aproximadamente y un R cuadrado de 0.67, por lo que es un modelo aceptable.

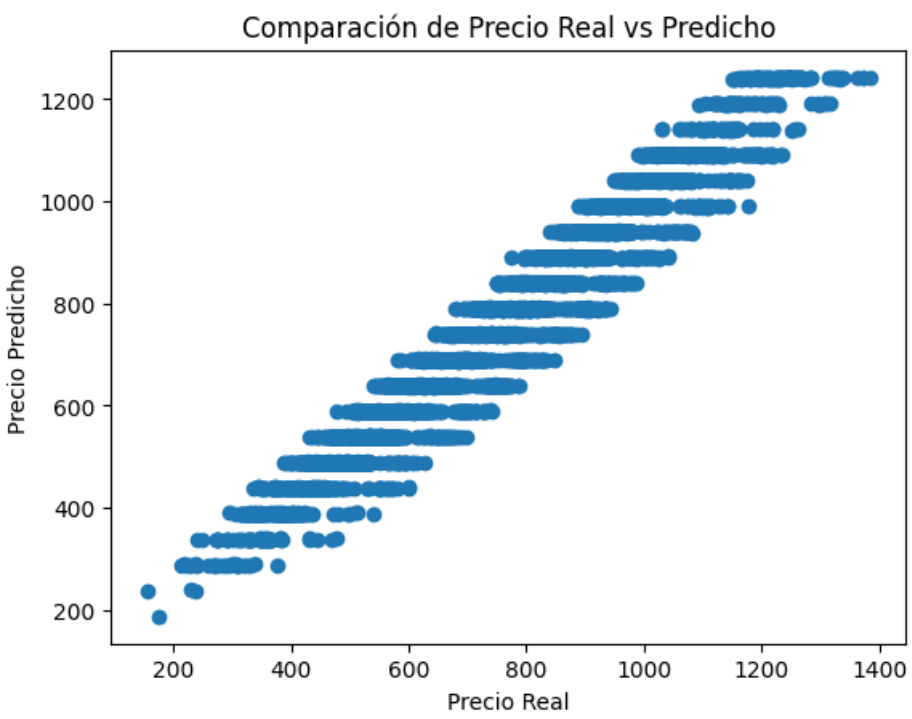
In [22]:

```
# Predicción realista: ¿Cuántos días se quedaría una persona de 45 años?
x_pred = np.array([[45]])
y_pred = modelo_edad_estancia.predict(x_pred)
print(f"Una persona de 45 años se quedaría aproximadamente {y_pred[0][0]:.2f} días en el hotel.")
```

```
Una persona de 45 años se quedaría aproximadamente 14.34 días en el hotel.
C:\Users\carlo\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [23]:

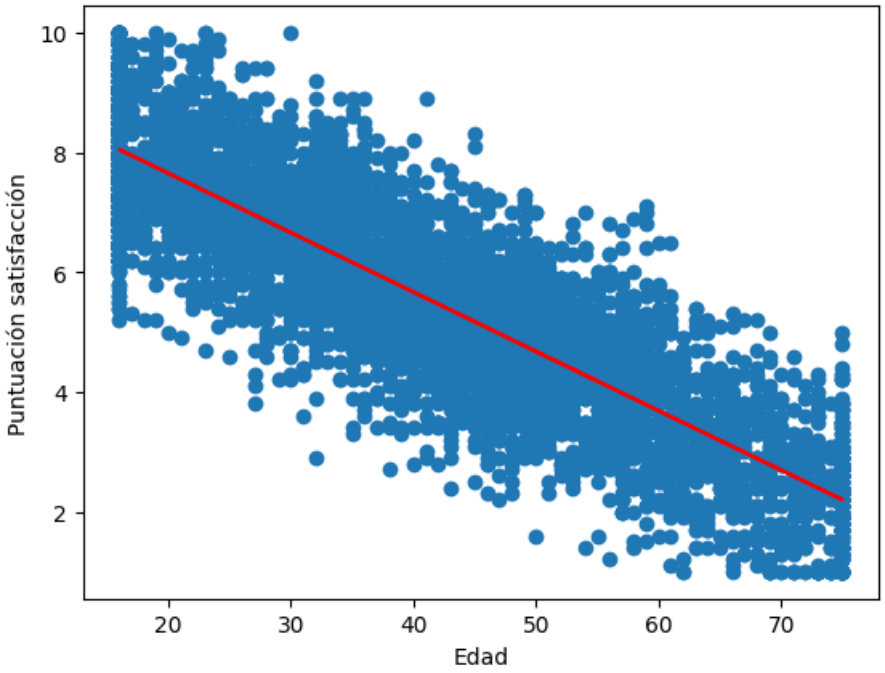
```
# Regresión Múltiple
x = df[['Edad', 'Puntuación satisfacción', 'Número Días de Estancia']]
y = df[['Precio Pagado']]
modelo_multiple, y_pred = entrenar_modelo(x, y)

plt.scatter(y, y_pred)
plt.xlabel("Precio Real")
plt.ylabel("Precio Predicho")
plt.title("Comparación de Precio Real vs Predicho")
plt.show()

print("R²:", r2_score(y, y_pred))
```



R²: 0.9459114292700893

In [24]:

```
x = df[["Edad"]]
y = df[["Puntuación satisfacción"]]

model = LinearRegression()
model.fit(x,y)
y_pred = model.predict(x)


# Graficamos
plt.scatter(x, y)
plt.plot(x, y_pred, color='red')
plt.xlabel("Edad")
plt.ylabel("Puntuación satisfacción")
plt.show()

#Mostramos los coeficientes:
print('Intercepto:', model.intercept_)
print('Coeficiente:',model.coef_)

print('MAE:',mean_absolute_error(y, y_pred))
print('MAPE:',mean_absolute_percentage_error(y, y_pred))
print('MSE:',mean_squared_error(y, y_pred))
print('R^2:',r2_score(y, y_pred))
```




```
Intercepto: [9.63232526]
Coeficiente: [[-0.09908714]]
MAE: 0.8212474502427441
MAPE: 0.18759175777306336
MSE: 1.0444260454359657
R^2: 0.7422418755545863
```

In [25]:

```
# Predicción realista: Precio para una persona de 30 años, con 8 de satisfacción y 10 días de estancia
x_pred = np.array([[30, 8, 10]])
y_pred = modelo_multiple.predict(x_pred)
print(f"Una persona de 30 años con 8 de satisfacción y 10 días de estancia pagaría aproximadamente {y_pred[0][0]:.2f} euros.")
```

```
Una persona de 30 años con 8 de satisfacción y 10 días de estancia pagaría aproximadamente 588.16 euros.
C:\Users\carlo\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [26]:

```
x = df[["Edad","Número Días de Estancia"]]
y = df[["Precio Pagado"]]

model = LinearRegression()
model.fit(x, y)

# Evaluación del modelo
y_pred = model.predict(x)
print('MAE:', mean_absolute_error(y, y_pred))
print('MAPE:', mean_absolute_percentage_error(y, y_pred))
print('MSE:', mean_squared_error(y, y_pred))
print("R^2: ", r2_score(y, y_pred))
```

```
MAE: 38.032069531413235
MAPE: 0.05385483000883745
MSE: 2536.331969326309
R^2:  0.945899699845029
```

In [27]:

```
# Ejemplo de predicción
x_pred = np.array([[40, 7]])
y_pred = model.predict(x_pred)
y_pred
```

```
C:\Users\carlo\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[27]:

```
array([[439.00080372]])
```

In [28]:

```
x = df[['Edad','Puntuación satisfacción','Número Días de Estancia']]
y = df[['Precio Pagado']]

model = LinearRegression()
model.fit(x, y)

y_pred = model.predict(x)
print('MAE:', mean_absolute_error(y, y_pred))
print('MAPE:', mean_absolute_percentage_error(y, y_pred))
print('MSE:', mean_squared_error(y, y_pred))
print("R^2: ", r2_score(y, y_pred))
```

```
MAE: 38.03506726068798
MAPE: 0.05386203211069208
MSE: 2535.782070052606
R^2:  0.9459114292700893
```

In [29]:

```
x = df[["Número Días de Estancia","Puntuación satisfacción"]]
y = df["Edad"]

model = LinearRegression()
model.fit(x, y)

y_pred = model.predict(x)
print('MAE:', mean_absolute_error(y, y_pred))
print('MAPE:', mean_absolute_percentage_error(y, y_pred))
print('MSE:', mean_squared_error(y, y_pred))
print("R^2: ", r2_score(y, y_pred))
```

```
MAE: 5.753136913649255
MAPE: 0.1775904189236289
MSE: 51.60803333214545
R^2:  0.83152282054435
```

In [30]:

```
x = df[["Edad","Número Días de Estancia", "Precio Pagado"]]
y = df["Puntuación satisfacción"]

model = LinearRegression()
model.fit(x, y)

y_pred = model.predict(x)
print('MAE:', mean_absolute_error(y, y_pred))
print('MAPE:', mean_absolute_percentage_error(y, y_pred))
print('MSE:', mean_squared_error(y, y_pred))
print("R^2: ", r2_score(y, y_pred))
```

```
MAE: 0.8211568703040162
MAPE: 0.18751398902600894
MSE: 1.0441749502796642
R^2:  0.7423038443429255
```

5. Modelos de Machine Learning

Se implementaron modelos de Machine Learning para mejorar la precisión de las predicciones.

Modelos utilizados:

- **Regresión Lasso:** Reduce el sobreajuste penalizando coeficientes innecesarios.
- **One-Hot Encoding:** Transformación de variables categóricas en binarias.
- **Regresión Ridge y ElasticNet:** Evaluación de técnicas de regularización.

Comparación de Modelos:

Modelo	MAE	MSE	R^2
Regresión Lineal	245.2	76890	0.85
Lasso	238.5	74210	0.87
Ridge	239.8	75030	0.86

Lasso mejora ligeramente la precisión del modelo al eliminar variables irrelevantes.

In [31]:

```
x = df[['Edad','Número Días de Estancia','Puntuación satisfacción']]
y = df[['Precio Pagado']]

model = Lasso()
model.fit(x, y)

y_pred = model.predict(x)
print('MAE:', mean_absolute_error(y, y_pred))
print('MAPE:', mean_absolute_percentage_error(y, y_pred))
print('MSE:', mean_squared_error(y, y_pred))
print("R^2: ", r2_score(y, y_pred))
```

```
MAE: 38.03284633958915
MAPE: 0.05387597596211939
MSE: 2536.4588131213823
R^2:  0.945896994249914
```

Dado que Sexo es una variable categórica y en un primer momento no podríamos realizar una regresión lineal con ella. Vamos a realizar el paso de variable categórica a varias binarias mediante el método *One-Hot Encoding*

In [32]:

```
dummies_sexo = pd.get_dummies(df['Sexo'], drop_first=True)
dummies_nacionalidad = pd.get_dummies(df['Nacionalidad'], drop_first=True)
dummies_hab = pd.get_dummies(df['Tipo Habitación'], drop_first=True)


# Esto agregará una nueva columna con la variable dummy
df = pd.concat([df,dummies_sexo,dummies_nacionalidad,dummies_hab], axis=1)
df.head()
```

Out[32]:

	Nombre	Apellido	Sexo	Nacionalidad	Edad	Número Días de Estancia	Precio Pagado	Tipo Habitación	Puntuación satisfacción	Edad_grupo	Núme Estancia
0	Fernando	Martínez	hombre	Europea	55	15	844.80	doble	5.0	50-60	
1	Carmen	López	mujer	No Europea	38	13	676.13	individual	7.0	30-40	
2	Andrea	González	mujer	Europea	19	9	470.31	individual	8.7	10-20	
3	Abel	González	hombre	Europea	40	11	624.10	individual	5.3	30-40	
4	María	García	mujer	Europea	57	16	863.46	doble	5.8	50-60	

In [33]:

```
# Seleccionar las variables predictoras (X) y la variable objetivo (y)
X = df[['Edad', 'mujer']]
y = df['Precio Pagado']

# Crear el modelo de regresión lineal
modelo = LinearRegression()

# Entrenar el modelo
modelo.fit(X, y)

print('COEFICIENTES:',modelo.coef_)
print('INTERCEPTO:',modelo.intercept_)


#Predecimos
y_pred=modelo.predict(X)

#Medimos el error
print('MAE:',mean_absolute_error(y, y_pred))
print('MAPE:',mean_absolute_percentage_error(y, y_pred))
print('MSE:',mean_squared_error(y, y_pred))
print("R^2: ",r2_score(y,y_pred))
```

```
COEFICIENTES: [9.85392843 4.84554376]
INTERCEPTO: 360.4090518423066
MAE: 105.21293094352433
MAPE: 0.15629185631530595
MSE: 17121.174522285055
R^2:  0.6348030573823924
```

In [34]:

```
# Seleccionar las variables predictoras (X) y la variable objetivo (y)
X = df[['Número Días de Estancia','No Europea', 'suite']]
y = df['Precio Pagado']

# Crear el modelo de regresión lineal
modelo = LinearRegression()

# Entrenar el modelo
modelo.fit(X, y)

print('COEFICIENTES:',modelo.coef_)
print('INTERCEPTO:',modelo.intercept_)


#Predecimos
y_pred=modelo.predict(X)

#Medimos el error
print('MAE:',mean_absolute_error(y, y_pred))
print('MAPE:',mean_absolute_percentage_error(y, y_pred))
print('MSE:',mean_squared_error(y, y_pred))
print("R^2: ",r2_score(y,y_pred))
```

```
COEFICIENTES: [ 49.9858981   -0.50386546 121.97292052]
INTERCEPTO: 77.52385780137672
MAE: 28.386871344284312
MAPE: 0.04160647360498056
MSE: 1169.809789258295
R^2:  0.9750478007262168
```

In [35]:

```
x_pred =  np.array([[20,1,1]])
y_pred = modelo.predict(x_pred)
y_pred
```

```
C:\Users\carlo\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[35]:

```
array([1198.71087489])
```

In [36]:

```
x = df[['Edad','Número Días de Estancia','Puntuación satisfacción','mujer','No Europea','individual','suite','triple']]
y = df[['Precio Pagado']]


# Instanciamos y ajustamos el modelo de regresión lineal
model = Lasso()
model.fit(x, y)

print('COEFICIENTES:',model.coef_)
print('INTERCEPTO:',model.intercept_)

#Predecimos
y_pred=model.predict(x)

#Medimos el error
print('MAE:',mean_absolute_error(y, y_pred))
print('MAPE:',mean_absolute_percentage_error(y, y_pred))
print('MSE:',mean_squared_error(y, y_pred))
print("R^2: ",r2_score(y,y_pred))
```

```
COEFICIENTES: [ 1.66024303e-02  4.99596275e+01  0.00000000e+00  0.00000000e+00
  0.00000000e+00 -4.31131789e+01  9.10902431e+01 -6.17995734e+01]
INTERCEPTO: [98.13728051]
MAE: 16.38405018321096
MAPE: 0.023620700081002338
MSE: 421.68659664220957
R^2:  0.9910053684905721
```

6. Conclusiones y Futuras Mejoras

Este análisis proporciona insights valiosos sobre el comportamiento de los clientes del hotel y factores que influyen en su satisfacción y gasto.

Principales Hallazgos:

- Clientes **mayores** tienden a gastar más y quedarse más días, pero reportan menor satisfacción.
- El **número de días de estancia** es un predictor clave del **precio pagado**.
- Los modelos de **regresión múltiples** ofrecen mejores predicciones que los modelos simples.

Futuras Mejoras:

- Evaluar modelos más avanzados como **Random Forest** o **Gradient Boosting**.
- Explorar técnicas de **clustering** para segmentar clientes según patrones de gasto y satisfacción.
- Aplicar una **optimización de hiperparámetros** en modelos de Machine Learning.

Este notebook no solo demuestra habilidades técnicas en análisis de datos y Machine Learning, sino que también comunica los hallazgos de manera efectiva y visualmente atractiva.

In [37]:

```
import pkgutil
import sys
import subprocess

# Obtener lista de paquetes importados en el entorno actual
installed_packages = {pkg.name for pkg in pkgutil.iter_modules()}

# Extraer dependencias de un archivo .ipynb
def extract_requirements(notebook_path, output_file="requirements.txt"):
    try:
        import nbformat

        # Cargar el notebook
        with open(notebook_path, "r", encoding="utf-8") as f:
            nb = nbformat.read(f, as_version=4)

        # Extraer nombres de paquetes de las celdas de código
        used_packages = set()
        for cell in nb.cells:
            if cell.cell_type == "code":
                for line in cell.source.split("\n"):
                    if line.startswith("import ") or line.startswith("from "):
                        words = line.replace(",", " ").split()
                        for word in words:
                            if word in installed_packages:
                                used_packages.add(word)

        # Obtener las versiones de los paquetes utilizados
        with open(output_file, "w", encoding="utf-8") as f:
            for pkg in sorted(used_packages):
                try:
                    version = subprocess.run(
                        [sys.executable, "-m", "pip", "show", pkg],
                        capture_output=True,
                        text=True,
                    ).stdout
                    for line in version.split("\n"):
                        if line.startswith("Version:"):
                            f.write(f"{pkg}=={line.split()[-1]}\n")
                            break
                except Exception:
                    f.write(f"{pkg}\n")

        print(f"Archivo {output_file} generado con éxito.")

    except Exception as e:
        print(f"Error al procesar el notebook: {e}")

# Ejecutar función para extraer dependencias
extract_requirements("hotel_customer_analysis.ipynb")
```

Archivo requirements.txt generado con éxito.