WEB SCRAPING CON SELENIUM EN BOOKING.COM Y VISUALIZACIÓN CON DASH

GESTION Y EXPLOTACIÓN DE ALMACENES DE DATOS

JOSELIN CALCI
ALEXANDRO BAZÁN

Tabla de contenido

WEB SCRAPING CON SELENIUM EN BOOKING.COM Y VISUALIZACIÓN CON DASH ***0***

Introducción ***3***

1. Metodología ***4***

2. Obtención de Datos mediante Web Scraping ***4***

2.1 Tecnologías Utilizadas **4**

2.2 Selección de la Fuente de Datos **5**

2.3 Implementación del Web Scraping **5**

2.4 Código Implementado **7**

2.5 Consideraciones Técnicas y Problemas Encontrados **11**

3. Limpieza y Transformación de Datos ***12***

3.1 Exploración Inicial y Detección de Problemas **12**

3.2 Procesos de Limpieza Aplicados **13**

3.3 Almacenamiento de Datos Limpios **17**

4. Análisis Exploratorio de Datos (EDA) ***18***

4.1 Estadística Descriptiva **18**

4.2 Distribución de Precios: **19**

4.3 Distribución de Puntuaciones **20**

4.4 Número de Reseñas **20**

4.5 Relación entre Precio y Estrellas **21**

4.6 Relación entre Puntuación y Precio **21**

4.7 Relación entre Número de Reseñas y Puntuación **22**

4.8 Correlación entre Variables **23**

4.9 Hoteles con Más Reseñas **23**

4.10 Comparación de Precios por Distrito **24**

..... 24

4.11 Distribución de Alojamiento por Distritos **25**

4.12 Ranking de Alojamientos **25**

4.13 Distribución de Estrellas **26**

4.14 Proporción de Alojamientos con Más de 1000 Reseñas **26**

5. Construcción del Dashboard Interactivo ***28***

5.1 Componentes del Dashboard: **28**

5.2 Tecnologías Utilizadas **29**

5.3 Implementación del Dashboard 29
5.4 Implementación de los Gráficos Interactivos 31
6. Dashboard Interactivo y Visualización de Datos 33
7. Conclusiones 42
8. Bibliografía 43

Introducción

En un mundo cada vez más digitalizado, la toma de decisiones basada en datos se ha convertido en una herramienta fundamental en diversas industrias. En el sector turístico y hotelero, la información sobre precios, ubicaciones, puntuaciones y reseñas de alojamientos resulta clave tanto para los viajeros como para los gestores de estos establecimientos.

En este contexto, el presente trabajo tiene como objetivo desarrollar un cuadro de mando interactivo que permita analizar la oferta de alojamientos en Madrid. Para ello, se ha utilizado la técnica de Web Scraping para extraer datos de Booking.com, seguido de un proceso de limpieza y transformación de datos, y finalmente la creación de un dashboard interactivo con Dash y Plotly, donde se presentan diversas visualizaciones para el análisis exploratorio de la información recopilada.

Fases del Proyecto
El desarrollo de este proyecto se ha dividido en las siguientes fases:

- Obtención de Datos: Web Scraping en Booking.com, extrayendo información clave como precios, puntuaciones, número de reseñas, ubicación y categoría de estrellas.
- Limpieza y Procesamiento de Datos: Tratamiento de valores nulos, eliminación de duplicados y transformación de los datos para asegurar su calidad y coherencia.
- Análisis Exploratorio de Datos (EDA): Visualización inicial de la distribución de precios, la relación entre puntuación y costo, y análisis de los alojamientos con mayor número de reseñas.
- Construcción del Dashboard Interactivo: Desarrollo de una plataforma visual con Dash y Plotly, incorporando gráficos como un mapa interactivo, histogramas y ranking de distritos.

Este análisis permitirá obtener información valiosa sobre la distribución de precios, la calidad de los alojamientos en diferentes distritos y la percepción de los usuarios basada en sus reseñas. Además, el dashboard resultante podrá ser utilizado como una herramienta de apoyo para viajeros, inversores o administradores del sector hotelero.

WEB SCRAPING CON SELENIUM EN BOOKING.COM Y VISUALIZACIÓN CON DASH

En el presente trabajo se ha desarrollado un proceso completo de obtención, limpieza y visualización de datos sobre alojamientos en Madrid a partir de la plataforma Booking.com. Dado que la web no ofrece una API pública para acceder a su información, se implementó Web Scraping utilizando Selenium para la extracción automatizada de datos. Posteriormente, se aplicó un proceso de limpieza y transformación con Pandas para garantizar la calidad de los datos. Finalmente, la información obtenida se representó de manera interactiva a través de un dashboard en Dash y Plotly, permitiendo el análisis dinámico de los alojamientos según su precio, puntuación, ubicación y otros atributos relevantes.

El presente informe detalla cada una de las etapas del proyecto, desde la recopilación de datos hasta la construcción del cuadro de mando, resaltando las técnicas utilizadas, los retos enfrentados y los hallazgos obtenidos.

1. Metodología

Para realizar el análisis de los alojamientos en Madrid disponibles en la plataforma Booking.com, se siguió un proceso estructurado que incluye la obtención de los datos mediante Web Scraping, su posterior limpieza y transformación, un Análisis Exploratorio de Datos (EDA) y finalmente, la visualización interactiva de la información a través de un dashboard en Dash. A continuación, se detallan cada una de las fases de la metodología aplicada.

2. Obtención de Datos mediante Web Scraping

2.1 Tecnologías Utilizadas

Para este trabajo, se utilizaron las siguientes herramientas:

Librería	Función
Selenium	Automatización de navegación y extracción de datos.
Pandas	Procesamiento y limpieza de datos.
Plotly	Creación de visualizaciones interactivas.
Dash	Desarrollo del cuadro de mando interactivo.

Para extraer información de Booking.com, se utilizó la técnica de Web Scraping mediante la biblioteca Selenium en Python. Esto permitió automatizar la navegación por la web y capturar datos clave sobre los alojamientos, tales como:

2.2 Selección de la Fuente de Datos

Booking.com es una de las plataformas de reservas más utilizadas en el mundo y ofrece información relevante sobre alojamientos, incluyendo su precio, puntuación, ubicación y número de reseñas. Sin embargo, Booking no cuenta con una API pública, lo que nos obligó a emplear Web Scraping para extraer estos datos.

Para lograrlo, utilizamos la biblioteca Selenium en Python, la cual permite automatizar la navegación en la web, interactuar con elementos de la página y extraer información de manera programada. Esto nos permitió simular el comportamiento de un usuario en el sitio web, desplazándonos por las páginas, realizando búsquedas y extrayendo datos estructurados.

2.3 Implementación del Web Scraping

El proceso de scraping en Booking.com se realizó siguiendo una serie de pasos clave:

a) **Inicialización del navegador con Selenium WebDriver:** Se configuró Google Chrome en modo headless (sin interfaz gráfica) para optimizar la velocidad del scraping.
b) **Acceso a la página principal de Booking:** Se utilizó Selenium para abrir la página y esperar a que el contenido se cargara correctamente.
c) **Automatización de la búsqueda de alojamientos en Madrid:** Se localizó la barra de búsqueda y se ingresó la palabra clave correspondiente.
d) **Extracción de información clave de los alojamientos:** Se recorrieron los resultados de la búsqueda y se extrajeron datos de: Una vez cargada la página con los resultados de búsqueda, se recorrieron los elementos HTML correspondientes a cada alojamiento y se extrajeron los siguientes datos:

- **Nombre del alojamiento**: Se identificó el nombre del hotel dentro del bloque de información de cada resultado.
- **Precio por noche**: Se localizó el precio en la estructura de la página y se procesó para eliminar caracteres adicionales (símbolos de moneda).
- **Puntuación promedio**: Se extrajo la calificación otorgada por los usuarios de Booking.

- **Número de reseñas**: Se obtuvo el número total de comentarios dejados por los visitantes.
- **Ubicación**: Se capturó la dirección del hotel, incluyendo su distrito dentro de Madrid.
- **Categoría de estrellas**: Se detectó la cantidad de estrellas del alojamiento, considerando que algunos pueden no tener esta información explícita.

Se verificó que los datos extraídos estuvieran correctamente alineados y correspondieran a los alojamientos adecuados. En caso de encontrar valores ausentes o mal formateados, se aplicaron ajustes para mantener la integridad del conjunto de datos.

e) **Navegación entre múltiples páginas de resultados**: Dado que Booking.com muestra los alojamientos en páginas, fue necesario recorrer varias páginas para recolectar una cantidad representativa de datos.

1. Se programó un script para detectar y hacer clic en el botón "Siguiente", permitiendo avanzar en la lista de resultados.
2. Se establecieron pausas entre cada navegación para evitar que el sitio detectara un comportamiento sospechoso y bloqueara el acceso.
3. Se iteró este proceso hasta alcanzar el número deseado de registros o hasta que no hubiera más páginas disponibles.

f) **Almacenamiento de los datos en un archivo CSV**: Una vez completada la extracción, los datos obtenidos fueron organizados en un DataFrame de Pandas y almacenados en un archivo CSV (alojamientos_booking.csv). También se generó una copia en formato Excel (alojamientos_booking.xlsx) para facilitar su visualización y análisis en herramientas externas.

Se verificó la calidad de los datos almacenados inspeccionando las primeras filas del DataFrame (df.head()), asegurando que cada columna contuviera la información esperada sin valores faltantes o desalineados.

Con este proceso, se obtuvo un conjunto de datos estructurado con 406 registros de alojamientos en Madrid, listo para ser limpiado y analizado en las siguientes etapas del proyecto.

2.4 Código Implementado

El código utilizado para realizar el scraping en **Booking.com** se organizó en los siguientes módulos:

Instalación de librerías necesarias

```
!pip install selenium pandas openpyxl
```

Automatización de la búsqueda en Booking

```
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import time

# Pedir al usuario que ingrese el destino
destino = input(" Ingresa el destino (ej. 'Sierra Norte de Madrid
'): ")

# Encontrar el campo de búsqueda del destino
search_box = driver.find_element(By.NAME, "ss")
search_box.clear()  # Limpiar el campo por si tiene texto previo
search_box.send_keys(destino)  # Escribir el destino

# Esperar a que la lista de sugerencias aparezca
time.sleep(2)

# Confirmar selección con ENTER
search_box.send_keys(Keys.RETURN)  # Presionar Enter nuevamente

print(f" Destino seleccionado: {destino}")

# Esperar a que la página de resultados cargue
time.sleep(5)
```

Configuración de Selenium y Web Scraping

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
import time

# Ruta correcta de ChromeDriver en Mac (ajusta si es necesario)
chrome_driver_path = r"/usr/local/bin/chromedriver"

# Configurar Selenium
service = Service(chrome_driver_path)
options = Options()
options.add_argument("--start-maximized")  # Maximizar ventana
```

```
# Inicializar el navegador
driver = webdriver.Chrome(service=service, options=options)

# URL de Booking con hoteles en Madrid
url = "https://www.booking.com/"
driver.get(url)

# Esperar 5 segundos para cargar la página
time.sleep(5)

print(" Página de Booking cargada correctamente")
```

Extraer nombres de alojamientos

⇒ **Extraer Precios**

```
# Extraer precios de los Alojamientos
price_elements = driver.find_elements(By.XPATH, '//span[@data-tes
tid="price-and-discounted-price"]')
prices = [price.text for price in price_elements]

print(f" Precios encontrados: {len(prices)}")
print("Ejemplo de precios:", prices[:10])
```

⇒ **Extraer Puntuación Especifica**

```
# Encontrar todos los bloques de alojamientos en la página
hotel_blocks = driver.find_elements(By.XPATH, '//div[@data-testid
="property-card"]')

# Lista para almacenar puntuaciones correctamente alineadas
ratings = []

# Recorrer cada alojamiento y extraer su puntuación específica
for hotel in hotel_blocks:
    try:
        # Buscar la puntuación dentro del bloque del hotel
        rating_element = hotel.find_element(By.XPATH, './/div[con
tains(@class, "abf093bdfe d86cee9b25")]')
        rating_text = rating_element.text.strip()

        # Extraer solo el número (ejemplo: "Puntuación: 8,8 " → "
8,8")
        rating = rating_text.split()[-1].replace(',', '.')  # Ree
mplazar coma por punto
    except:
        rating = "Sin puntuación"

    # Agregar la puntuación a la lista
    ratings.append(rating)

# Verificar la cantidad de puntuaciones extraídas
print(f" Puntuaciones encontradas: {len(ratings)}")
```

```
print("Ejemplo de puntuaciones:", ratings[:10])
```

⇒ **Extraer Ubicación**

```
# Extraer ubicaciones
location_elements = driver.find_elements(By.XPATH, '//span[@data-
testid="address"]')

# Obtener los textos de ubicación
locations = [location.text for location in location_elements]

print(f"📍 Ubicaciones encontradas: {len(locations)}")
print("Ejemplo de ubicaciones:", locations[:10])
```

⇒ **Extraer Reseñas**

```
# Lista para almacenar reseñas correctamente alineadas
num_reviews = []

# Recorrer cada hotel en los bloques extraídos (para asegurar que
la reseña corresponde al hotel correcto)
for hotel in hotel_blocks:
    try:
        # Buscar la reseña dentro del bloque del hotel
        review_element = hotel.find_element(By.XPATH, './/span[co
ntains(@class, "f45d8e4c32 d935416c47")]')
        review_text = review_element.text.strip()

        # Extraer solo el número de comentarios
        review = review_text.replace(".", "").replace(" comentari
os", "") + " comentarios"
    except:
        review = "No hay reseñas"

    # Agregar la reseña a la lista
    num_reviews.append(review)

# **Verificar si las reseñas están alineadas correctamente**
print(f" Reseñas encontradas: {len(num_reviews)}")
print(f" Hoteles encontrados: {len(hotel_blocks)}")

# **Asegurar que coincidan las longitudes**
if len(num_reviews) == len(hotel_blocks):
    print(" Las reseñas están alineadas correctamente con los hot
eles.")
else:
    print(" ERROR: Las reseñas y los hoteles NO tienen el mismo t
amaño.")

    # **Si hay un desfase, ajustar con "No hay reseñas"**
    while len(num_reviews) < len(hotel_blocks):
        num_reviews.append("No hay reseñas")
    while len(num_reviews) > len(hotel_blocks):
        num_reviews.pop()
```

```
    print(" Se ha ajustado la longitud de las reseñas para que co
incida con los hoteles.")
    print(f"Nuevo total de reseñas: {len(num_reviews)}")
```

⇒ **Extraer número de estrellas**

```
# Encontrar todos los bloques de Alojamientos en la página
hotel_blocks = driver.find_elements(By.XPATH, '//div[@data-testid
="property-card"]')

# Lista para almacenar estrellas correctamente alineadas
stars = []

# Recorrer cada hotel y extraer el número de estrellas
for hotel in hotel_blocks:
    try:
        # Buscar el contenedor de estrellas dentro del hotel
        stars_element = hotel.find_element(By.XPATH, './/div[@dat
a-testid="rating-stars"]')

        # Contar la cantidad de iconos de estrellas dentro del co
ntenedor
        num_stars  =  len(stars_element.find_elements(By.TAG_NAME,
"svg"))
    except:
        num_stars = 0  # Si hay error o no se encuentran estrella
s, asignar 0

    # Convertir 0 en "Sin estrellas"
    stars.append(num_stars if num_stars > 0 else "Sin estrellas")

# **Asegurar que la lista de estrellas tenga el mismo tamaño que
la de hoteles**
stars = stars + ["Sin estrellas"] * (len(hotel_names) - len(stars
))

# Verificar la cantidad de estrellas extraídas
print(f" Hoteles con estrellas detectadas: {len(stars)}")
print("Ejemplo de estrellas:", stars[:10])
```

Creación y Almacenamiento de Datos Extraídos

```
import pandas as pd

# Crear DataFrame con los datos extraídos
df_hotels = pd.DataFrame({
    "Alojamiento": alojamientos,  # Cambio de "Hotel" a "Alojamie
nto"
    "Precio": prices,
    "Puntuación": ratings,
    "Reseñas": num_reviews,
    "Ubicación": locations,
    "Estrellas": stars  # Agregar estrellas
```

```
})

# Guardar en CSV
csv_filename = "alojamientos_booking.csv"
df_hotels.to_csv(csv_filename, index=False, encoding="utf-8")

# Guardar en Excel
excel_filename = "alojamientos_booking.xlsx"
df_hotels.to_excel(excel_filename, index=False)

# Mostrar los primeros registros para verificar
print(f" Archivos guardados:\n- {csv_filename}\n- {excel_filename
}")
print(f" Datos extraídos de {len(df_hotels)} alojamientos y guard
ados correctamente.")
print(df_hotels.head(10))
```

2.5 Consideraciones Técnicas y Problemas Encontrados

Durante el proceso de scraping en Booking.com, se identificaron varios desafíos que se resolvieron con diferentes técnicas:

Problema	Solución Implementada
Booking bloquea el acceso de bots	Uso de **pausas aleatorias** entre solicitudes para evitar detección anti-bot.
El contenido de la web carga dinámicamente	Uso de **Selenium WebDriver** en lugar de BeautifulSoup, ya que Selenium permite interactuar con elementos dinámicos.
Navegación entre páginas era limitada	Se automatizó la navegación con Selenium haciendo clic en el botón "Siguiente".
Algunos valores numéricos estaban en texto	Se realizó **conversión de datos** con Pandas para asegurar consistencia en los valores.

El proceso de Web Scraping permitió extraer y almacenar datos de alojamientos en Booking.com con éxito. Este conjunto de datos se utilizará en las siguientes etapas para su procesamiento, análisis exploratorio y visualización en un dashboard interactivo.

Resultados obtenidos:

Los datos obtenidos mediante web scraping en la plataforma Booking.com corresponden a alojamientos disponibles en Madrid. El conjunto de datos incluye información clave sobre cada alojamiento, como su nombre, precio por noche,

puntuación promedio, número de reseñas, ubicación y categoría de estrellas. Se extrajeron un total de 406 registros de alojamientos, los cuales fueron almacenados en formatos CSV y Excel para facilitar su posterior análisis. Por ejemplo, se han registrado alojamientos como "Hotel Fenix Gran Meliá", que tiene una puntuación de 8.8 basada en 660 reseñas, y un precio por noche de €7,809. Además, algunos alojamientos no presentan una categoría de estrellas explícita en la plataforma, por lo que estos valores aparecen como "Sin estrellas".

Los datos extraídos han sido preparados para su limpieza y transformación, asegurando su coherencia y calidad antes del análisis. Posteriormente, se llevará a cabo una exploración de tendencias y relaciones en los datos, permitiendo identificar patrones en la distribución de precios y evaluar la relación entre la calidad percibida y el costo de los alojamientos. Finalmente, los resultados obtenidos serán presentados en un Dashboard interactivo, desarrollado con Dash y Plotly, proporcionando visualizaciones dinámicas que facilitarán la interpretación y el análisis de la oferta hotelera en Madrid.

3. Limpieza y Transformación de Datos

Después de realizar el Web Scraping y obtener los datos de Booking.com, se identificaron diversas inconsistencias y ruidos en la información extraída. Debido a la estructura de los datos en la web, fue necesario aplicar una serie de procesos de limpieza y transformación para normalizar y estructurar la información de manera adecuada antes del análisis y la visualización.

3.1 Exploración Inicial y Detección de Problemas

Antes de comenzar con la limpieza de los datos, se realizó una **exploración inicial** para identificar posibles problemas en el conjunto de datos. Las principales inconsistencias detectadas fueron:

- **Formato incorrecto en precios**: Los valores de la columna "Precio" incluían el símbolo "€" y puntos como separadores de miles, lo que impedía realizar operaciones matemáticas.
- **Puntuaciones en formato de texto**: Algunas puntuaciones estaban almacenadas como cadenas de texto, en lugar de valores numéricos.
- **Inconsistencias en el número de reseñas**: Los datos de la columna "Reseñas" incluían caracteres no numéricos como "·" o la palabra "comentarios", lo que dificultaba su conversión a formato numérico.

- **Ubicaciones combinadas en una sola columna**: La ubicación de los alojamientos aparecía en formato "Distrito, Ciudad", lo que requería una separación en dos columnas distintas.
- **Valores faltantes en algunas columnas**: Había registros sin información de puntuaciones o estrellas, por lo que fue necesario imputar estos valores.

3.2 Procesos de Limpieza Aplicados

A continuación, se describen los pasos implementados para limpiar y transformar el conjunto de datos.

Importación de librerías

```
import pandas as pd
import numpy as np
# Cargar el dataset extraído de Booking
df = pd.read_csv("alojamientos_booking.csv")

# Mostrar las primeras filas
print("🔍 Primeras filas del DataFrame:")
print(df.head())
```

a) Conversión de precios a formato numérico

Los precios extraídos incluían el símbolo "€" y estaban en formato de texto. Para normalizar estos valores:

- Se eliminó el símbolo "€" y los separadores de miles "." y ",".
- Se aplicó pd.to_numeric() para convertir la columna a formato numérico.
- Se identificaron casos en los que los precios estaban mal escalados (mayores a 10,000), dividiéndolos por 100 para corregir la escala.

Código Implementado:

```
import pandas as pd

# Cargar los datos sin conversión automática
df = pd.read_csv("alojamientos_booking.csv", dtype=str)

# 1 **Eliminar símbolos y espacios en 'Precio'**
df["Precio"] = df["Precio"].str.replace("€", "").str.replace("
.", "").str.replace(",", "").str.strip()

# 2 **Convertir a número**
df["Precio"] = pd.to_numeric(df["Precio"], errors="coerce")
```

```
# 3 **Corregir valores que están mal escalados**
df["Precio"] = df["Precio"].apply(lambda x: x / 100 if x > 100
00 else x)

# **Verificar los primeros resultados**
print("\n Precios correctamente normalizados:")
print(df[['Alojamiento', 'Precio']].head(10))
```

b) Conversión de puntuaciones a valores numéricos

Los datos de puntuación estaban almacenados como texto, lo que impedía realizar análisis numéricos. Se aplicó:

- pd.to_numeric() para convertirlos a valores numéricos.
- Se reemplazaron los valores NaN por 0.0 en los casos en los que no existía una puntuación.

```
#Convertir puntuaciones a valores numéricos
df['Puntuación'] = pd.to_numeric(df['Puntuación'], errors='coe
rce')
# Reemplazar NaN por 0.0 sin 'chained assignment'
df.loc[df['Puntuación'].isna(), 'Puntuación'] = 0.0

# Mostrar primeros valores para verificar
print("\n Puntuaciones convertidas a valores numéricos:")
print(df[['Alojamiento', 'Puntuación']].head())
```

c) Normalización del número de reseñas

Los datos de reseñas contenían caracteres adicionales como "·" y la palabra "comentarios". Para corregir esto:

- Se eliminaron estos caracteres.
- Se convirtieron los valores a enteros (int), imputando 0 en los casos donde no había información.

```
# Eliminar el carácter "·" y la palabra "comentarios"
df['Reseñas'] = df['Reseñas'].astype(str).str.replace("·", "",
regex=True)
df['Reseñas'] = df['Reseñas'].str.replace(" comentarios", "", r
egex=True)

# Convertir a números enteros, tratando valores nulos como 0
```

```
df['Reseñas'] = pd.to_numeric(df['Reseñas'], errors='coerce').
fillna(0).astype(int)

# Mostrar primeros valores
print("\n Reseñas convertidas a valores numéricos:")
print(df[['Alojamiento', 'Reseñas']].head())
```

d) Separación de la ubicación en "Distrito" y "Ciudad"

La columna de ubicación contenía los valores en el formato "Distrito, Ciudad". Para normalizarlo:

- Se utilizó .str.split(',', expand=True) para separar los valores en dos columnas.
- Se limpiaron los espacios en blanco con .str.strip().
- Se estableció "Madrid" como valor predeterminado en los casos donde la ciudad no estaba especificada.

```
#Separar la columna de Ubicación en Distrito y Ciudad
df[['Distrito', 'Ciudad']] = df['Ubicación'].str.split(',', ex
pand=True)

# Limpiar espacios en blanco en ambas columnas
df['Distrito'] = df['Distrito'].str.strip()
df['Ciudad'] = df['Ciudad'].str.strip()

# Rellenar valores faltantes en Ciudad con "Madrid" si Distrit
o ya es "Madrid"
df['Ciudad'] = df['Ciudad'].fillna("Madrid")
df.loc[df['Distrito'] == "Madrid", 'Ciudad'] = "Madrid"

# Mostrar primeros valores
print("\n Ubicaciones normalizadas (Distrito y Ciudad corregid
os):")
print(df[['Alojamiento', 'Distrito', 'Ciudad']].head(15))
```

e) Manejo de valores faltantes en la columna "Estrellas"

Algunos alojamientos no tenían información sobre su categoría en estrellas, por lo que:

- Se reemplazó el valor "Sin estrellas" por 0.
- Se convirtió la columna a valores numéricos (float).

```
# Reemplazar "Sin estrellas" por 0 y convertir a número
df['Estrellas'] = df['Estrellas'].replace("Sin estrellas", 0).
astype(float)

# Mostrar primeros valores
print("\n Estrellas normalizadas:")
print(df[['Alojamiento', 'Estrellas']].head())
```

f) Análisis de valores faltantes

Se realizó un análisis para verificar si aún existían valores nulos en el conjunto de datos. Se generó un resumen con la cantidad de valores faltantes en cada columna.

Código Implementado:

```
import pandas as pd

# Crear DataFrame con información de los valores faltantes
df_info = pd.DataFrame({
    "Columnas": df.columns,
    "Datos Faltantes": df.isnull().sum().values,
    "Tipo de Dato": df.dtypes.values
})

# Aplicar estilos para mejor visualización en Jupyter Notebook
df_info_styled = df_info.style.set_table_styles([
    {"selector": "thead th", "props": "background-color: #0035
80; color: white; font-weight: bold; text-align: center;"},
    {"selector": "tbody td", "props": "text-align: center;"}]
).set_caption("🔍 Resumen de Datos Faltantes")

# Mostrar en Jupyter Notebook
display(df_info_styled)
```

Resumen de Datos Faltantes			
0	Alojamiento	0	object
1	Precio	0	float64
2	Puntuación	0	float64
3	Reseñas	0	int64
4	Ubicación	0	object
5	Estrellas	0	float64
6	Distrito	0	object
7	Ciudad	0	object

Según el resumen de Datos Faltantes muestra que no existen valores nulos en el conjunto de datos final, lo que indica que todas las columnas han sido correctamente procesadas y completadas. Se realizaron imputaciones y transformaciones necesarias para garantizar la coherencia y calidad de la información.

En particular, se detectaron valores ausentes en la columna "Estrellas", los cuales fueron reemplazados por 0, indicando que algunos alojamientos no cuentan con una clasificación específica. Además, se verificó que todas las demás variables, como "Precio", "Puntuación" y "Reseñas", están correctamente almacenadas en formato numérico (float64 o int64), lo que facilita su posterior análisis.

Gracias a este proceso de depuración, los datos se encuentran estructurados y listos para ser utilizados en el análisis exploratorio y visualización en el dashboard, evitando problemas de inconsistencia y garantizando la precisión de los resultados obtenidos.

3.3 Almacenamiento de Datos Limpios

Una vez finalizada la limpieza, el conjunto de datos fue guardado en dos formatos: CSV y Excel, para su posterior análisis y visualización.

Código Implementado:

```
#Guardar los datos limpios en CSV y Excel
csv_filename = "alojamientos_booking_limpio.csv"
excel_filename = "alojamientos_booking_limpio.xlsx"

df.to_csv(csv_filename, index=False, encoding="utf-8")
df.to_excel(excel_filename, index=False)

# Mensaje de confirmación
print(f"📁 Datos limpios guardados en:\n- {csv_filename}\n- {exce
l_filename}
```

Resultado: Tras aplicar el proceso de limpieza y transformación, se obtuvo un conjunto de datos estructurado y optimizado para su análisis, con un total de 406 registros de alojamientos en Madrid. La información ha sido normalizada, eliminando inconsistencias como símbolos en los precios, valores nulos en las puntuaciones y datos desalineados en las reseñas y ubicaciones.

Además, la segmentación de la columna "Ubicación" en "Distrito" y "Ciudad" ha permitido una mejor organización de los datos, facilitando el análisis geográfico de los alojamientos. Finalmente, los datos han sido almacenados en los archivos alojamientos_booking_limpio.csv y alojamientos_booking_limpio.xlsx, asegurando su accesibilidad para futuras visualizaciones y análisis exploratorios.

Este conjunto de datos limpio servirá como base para la construcción del dashboard interactivo, donde se explorarán patrones y tendencias clave en la oferta de alojamientos en Madrid.

4. Análisis Exploratorio de Datos (EDA)

Antes de construir el dashboard interactivo, se llevó a cabo un Análisis Exploratorio de Datos (EDA) para comprender mejor la distribución de las variables clave y detectar patrones relevantes en la oferta de alojamientos en Madrid.

Este análisis permitió identificar tendencias en los precios, puntuaciones, número de reseñas, categorización de los alojamientos por estrellas y su distribución por distrito.

4.1 Estadística Descriptiva

Para comenzar, se generó un resumen estadístico de las principales variables del conjunto de datos:

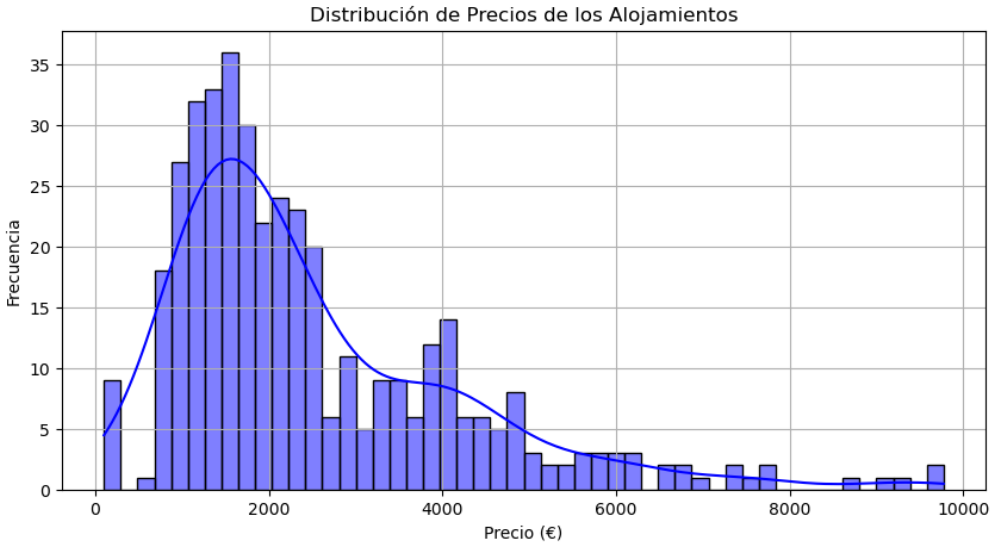
Estadísticos	Precio (€)	Puntuación	Reseñas	Estrellas
Total de registros	406	406	406	406
Media	2509.71	7.34	1662.49	1.45
Desviación estándar	1694.54	2.25	2372.50	1.86
Mínimo	101.17	0.00	0	0
Percentil 25%	1342.50	6.80	23	0
Mediana (50%)	1992.50	8.20	481.50	0
Percentil 75%	3348.25	8.70	2565	3
Máximo	9771.00	10.00	14779	5

- La media de los precios de los alojamientos es de 2509.71€, aunque la alta desviación estándar sugiere una gran dispersión de valores.
- Las puntuaciones promedian 7.34, indicando una percepción generalmente positiva de los alojamientos.
- Algunas propiedades tienen cero estrellas, lo que implica una gran presencia de apartamentos turísticos sin categorización oficial.

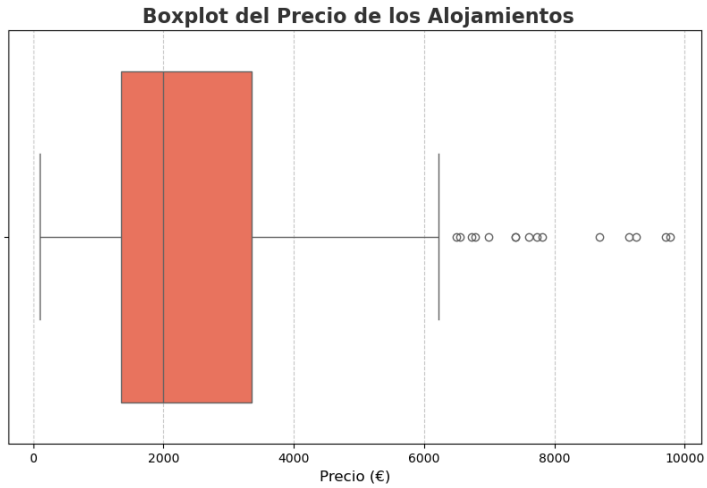
- Existen alojamientos con hasta 14,779 reseñas, lo que resalta su popularidad y alta ocupación.

4.2 Distribución de Precios:

El histograma de distribución de precios de los alojamientos muestra una gran variabilidad en los costos. Se observa una concentración de precios en el rango de 1000€ a 3000€, con una cola larga hacia valores más altos. Esto indica que la mayoría de los alojamientos se encuentran en un rango de precio accesible, mientras que existen algunos establecimientos de lujo con precios significativamente más elevados.

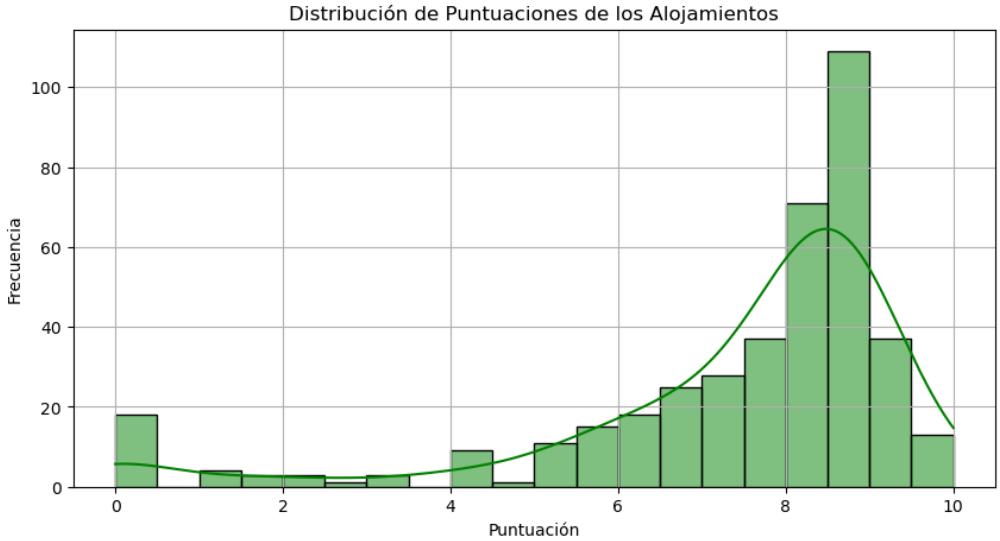


El boxplot complementa este análisis, revelando la presencia de outliers (valores atípicos) que superan los 6000€. Estos puntos corresponden a alojamientos exclusivos o de alta gama, lo que refleja la diversidad de opciones disponibles en la ciudad. La mediana del precio está en 1992.50€, reflejando el costo típico en la ciudad.



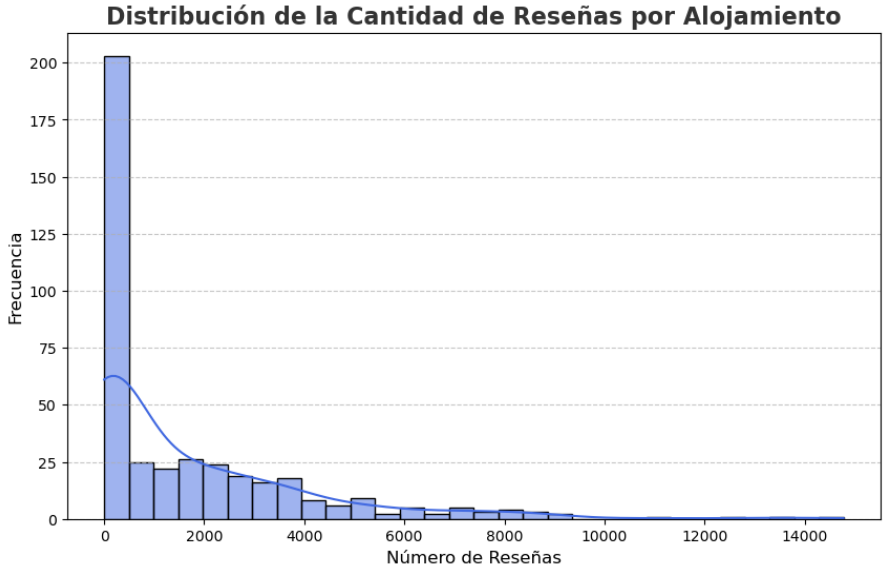
4.3 Distribución de Puntuaciones

El histograma de puntuaciones revela que la mayoría de los alojamientos tienen valoraciones superiores a 7.0, con un pico en 8.0 y 9.0. Esto sugiere que los alojamientos en Madrid suelen estar bien valorados por los usuarios, aunque también hay un pequeño grupo de alojamientos con puntuaciones cercanas a 0 o 1, lo que indica experiencias negativas.



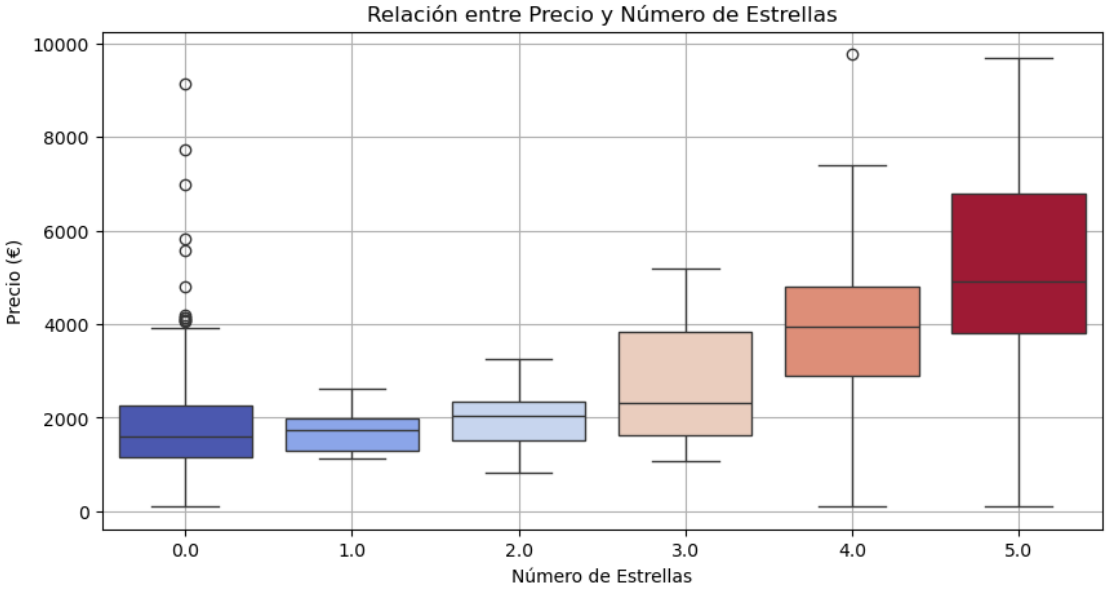
4.4 Número de Reseñas

Se analizó la cantidad de reseñas por alojamiento, observándose que una gran proporción de alojamientos tiene pocas reseñas (menores a 500), mientras que unos pocos destacan con más de 10,000 comentarios. Este patrón sugiere que los alojamientos más populares concentran una mayor cantidad de opiniones, mientras que la mayoría de los establecimientos tienen menor visibilidad o menor volumen de clientes.



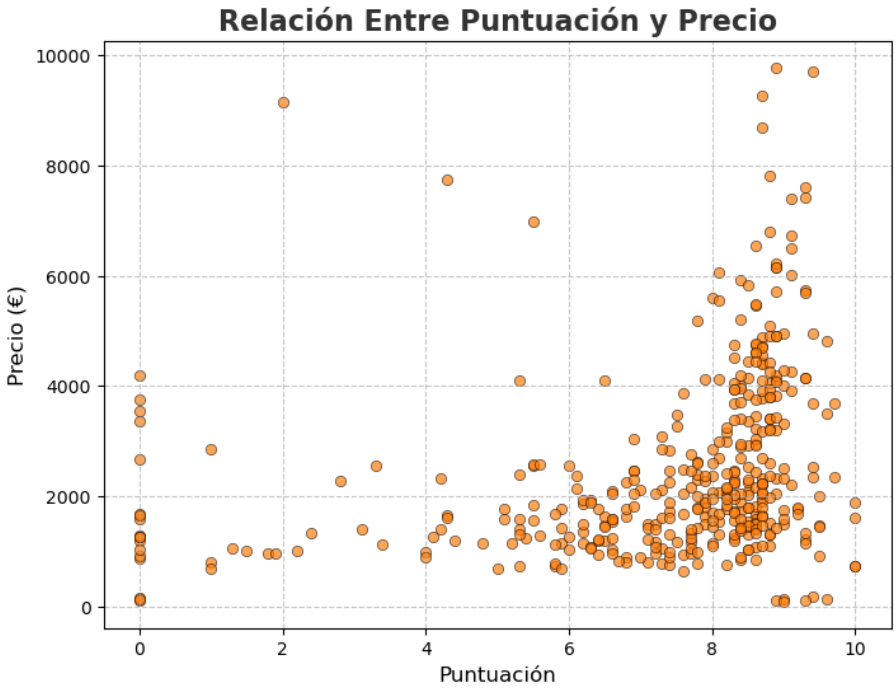
4.5 Relación entre Precio y Estrellas

El boxplot de la relación entre el número de estrellas y el precio confirma que los alojamientos con mayor categoría tienden a ser más costosos. Los alojamientos de 5 estrellas tienen precios significativamente más altos, con una mediana que supera los 4000€, mientras que los de 1 y 2 estrellas presentan precios más bajos y menos dispersión. Los alojamientos sin estrellas tienen una amplia variabilidad de precios, lo que indica que muchos apartamentos turísticos no siguen una estructura tarifaria estándar.



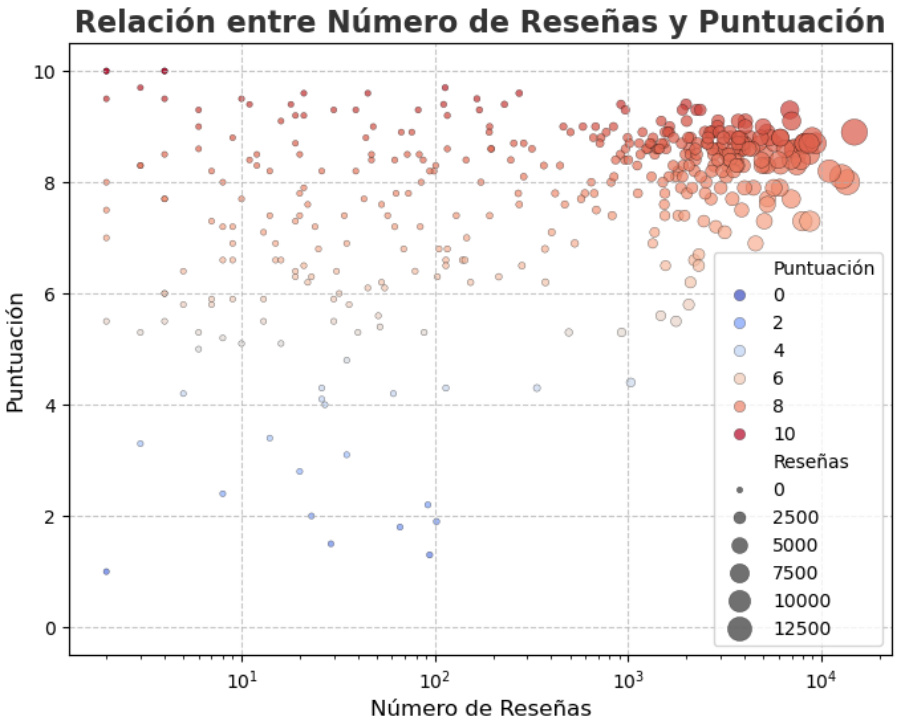
4.6 Relación entre Puntuación y Precio

El gráfico de dispersión entre la puntuación y el precio indica que no existe una correlación fuerte entre ambas variables. Si bien algunos de los alojamientos más costosos tienen altas puntuaciones, también hay opciones económicas con valoraciones elevadas. Esto sugiere que la calidad percibida de un alojamiento no siempre está relacionada con su costo.



4.7 Relación entre Número de Reseñas y Puntuación

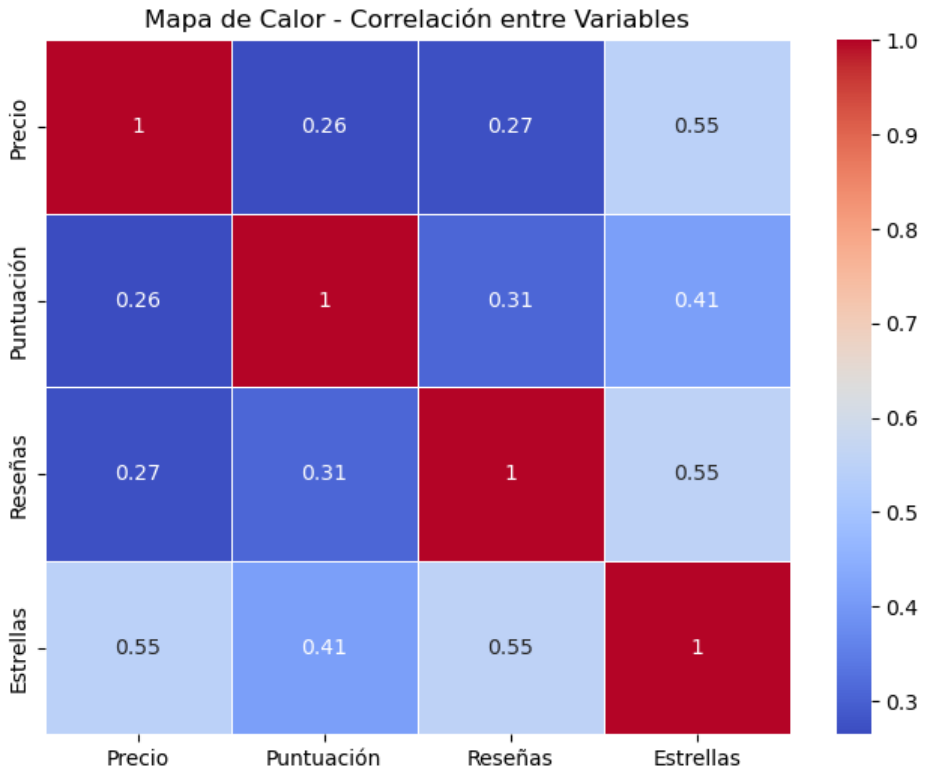
El gráfico de burbujas muestra que los alojamientos con más reseñas tienden a mantener puntuaciones altas, lo que sugiere que los establecimientos con mayor volumen de clientes suelen ofrecer experiencias satisfactorias. Sin embargo, algunos alojamientos con muchas reseñas presentan puntuaciones medianas o bajas, lo que podría deberse a una mayor exposición a críticas variadas.



4.8 Correlación entre Variables

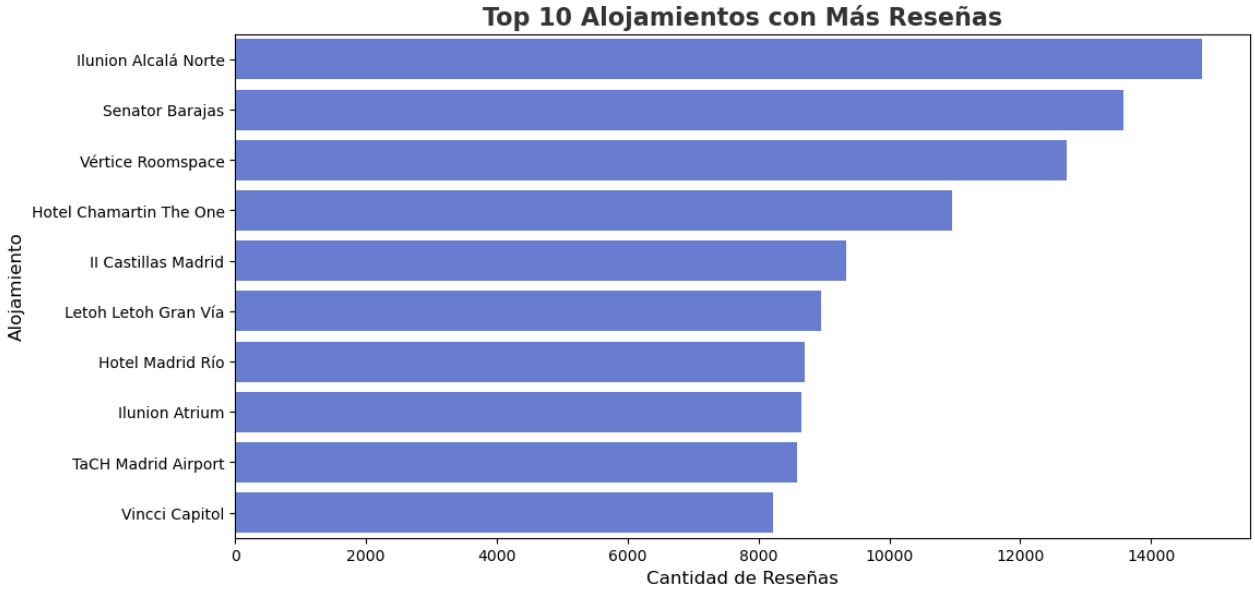
El mapa de calor de correlación muestra que:

- Existe una correlación moderada (0.55) entre el número de estrellas y el precio, lo que confirma que mayor categoría implica precios más altos.
- La relación entre puntuación y precio es baja (0.26), lo que indica que un mayor precio no siempre se traduce en una mejor valoración.
- El número de reseñas tiene una correlación moderada (0.55) con las estrellas, lo que sugiere que los alojamientos de mayor categoría suelen recibir más reseñas.



4.9 Hoteles con Más Reseñas

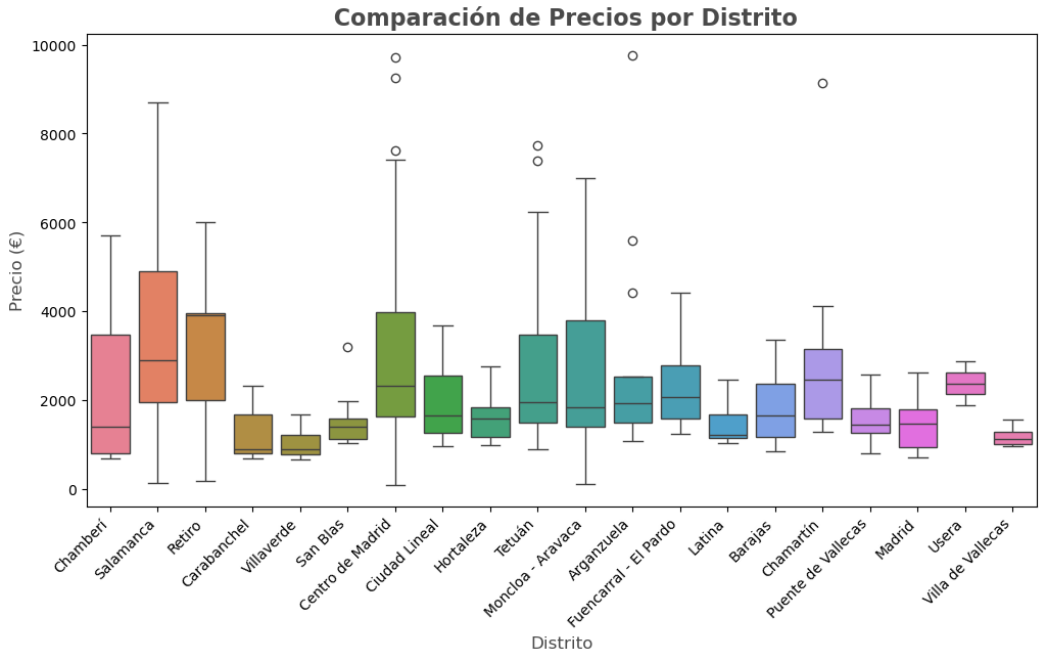
El análisis de los 10 alojamientos con más reseñas muestra que los establecimientos con mayor volumen de comentarios suelen ser hoteles reconocidos, lo que indica que la reputación y la presencia en plataformas como Booking influyen en la visibilidad de los alojamientos.



4.10 Comparación de Precios por Distrito

Se realizó un boxplot para comparar los precios de los alojamientos en cada distrito.

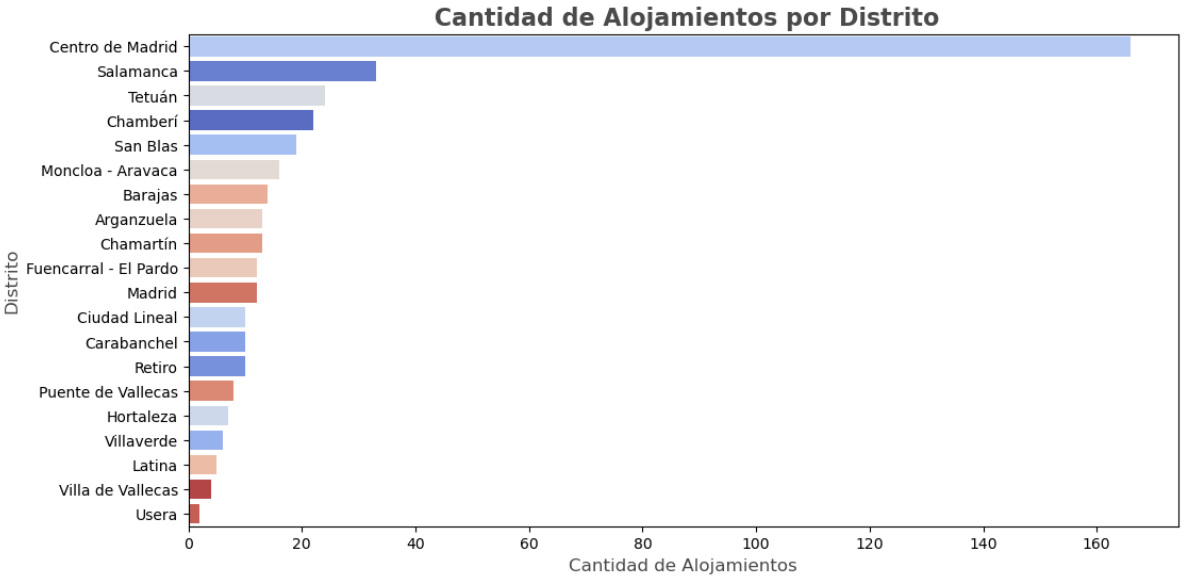
Los distritos de Salamanca, Centro de Madrid y Chamberí presentan los precios más elevados, lo que coincide con su reputación como zonas exclusivas de Madrid. En contraste, distritos como Puente de Vallecas, Usera y Villa de Vallecas tienen precios significativamente más bajos.



4.11 Distribución de Alojamiento por Distritos

Se identificó que la mayor cantidad de alojamientos se encuentra en el Centro de Madrid, seguido de Salamanca y Tetuán. Estos distritos presentan una mayor oferta de hospedajes debido a su atractivo turístico y comercial.

Además, el boxplot de precios por distrito evidencia que Salamanca y Retiro tienen los alojamientos más costosos en promedio, mientras que en distritos como Usera y Villa de Vallecas los precios son significativamente más bajos.



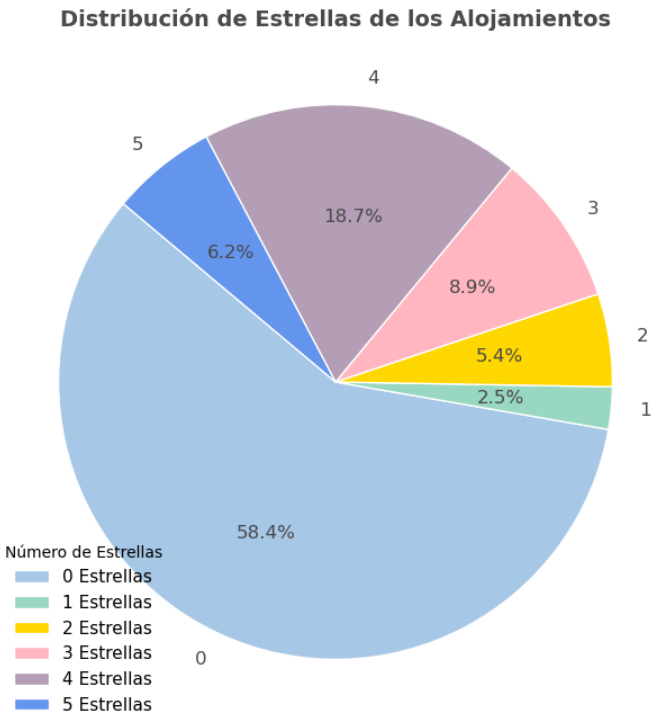
4.12 Ranking de Alojamientos

Finalmente, se generó un ranking de los Top 10 alojamientos mejor puntuados, donde se observa que la mayoría de ellos poseen una calificación cercana a 10, lo que indica altos estándares de calidad según la percepción de los huéspedes.

4.13 Distribución de Estrellas

Para visualizar la clasificación de los alojamientos según su número de estrellas, se generó un gráfico de pastel.

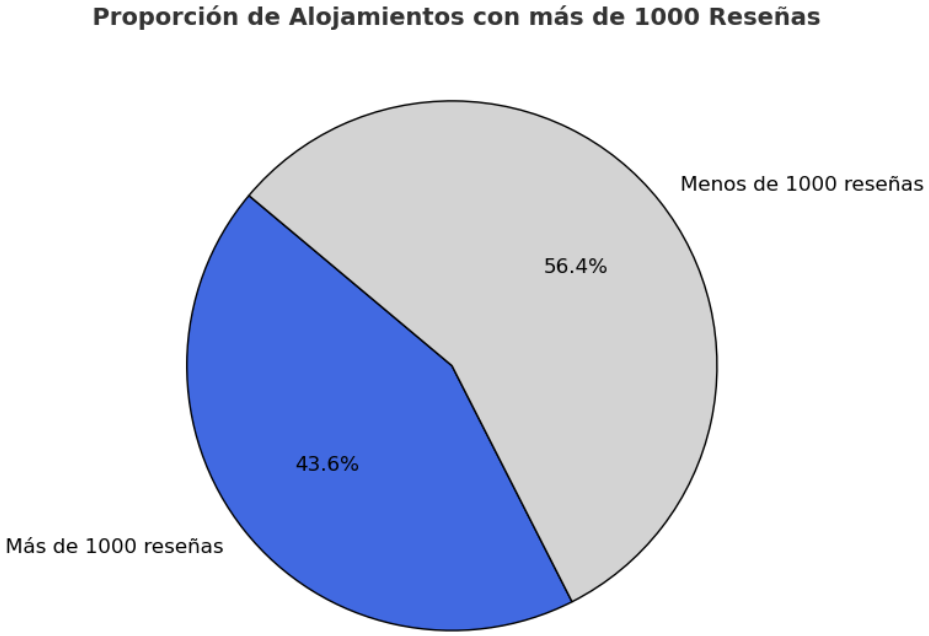
- Más del 58% de los alojamientos no tiene una clasificación de estrellas oficial, lo que indica una gran presencia de apartamentos y alojamientos turísticos.
- Los hoteles de 4 y 5 estrellas representan el 18.7% y 6.2% respectivamente, lo que sugiere que Madrid cuenta con una oferta significativa de alojamientos de lujo.



4.14 Proporción de Alojamientos con Más de 1000 Reseñas

Para conocer el porcentaje de alojamientos con una alta cantidad de reseñas, se generó un gráfico de pastel.

- El 43.6% de los alojamientos tiene más de 1000 reseñas, lo que indica un nivel alto de interacción por parte de los usuarios.
- El 56.4% tiene menos de 1000 reseñas, lo que podría estar asociado a alojamientos más nuevos o con menor visibilidad en la plataforma.



Resultados del EDA

El análisis exploratorio permitió obtener información valiosa sobre el mercado de alojamientos en Madrid:

- La mayoría de los alojamientos tienen precios entre 1000€ y 3000€, aunque existen opciones de lujo significativamente más costosas.
- Las puntuaciones generalmente son altas, reflejando una buena percepción de los alojamientos.
- No existe una correlación fuerte entre precio y puntuación, lo que indica que alojamientos más caros no necesariamente son mejor valorados.
- Los distritos más populares para alojarse son Centro, Salamanca y Tetuán.
- Los alojamientos con mayor número de reseñas tienden a tener una puntuación alta, lo que indica que la satisfacción de los clientes está relacionada con la cantidad de comentarios recibidos.

Este análisis proporciona una base sólida para la construcción del Dashboard Interactivo, donde los usuarios podrán explorar estas relaciones y tendencias de forma dinámica.

5. Construcción del Dashboard Interactivo

Para presentar de manera clara e interactiva los hallazgos obtenidos en el análisis exploratorio, se desarrolló un dashboard interactivo utilizando Dash y Plotly. Este dashboard permite explorar los datos dinámicamente, ofreciendo herramientas de filtrado y múltiples visualizaciones para facilitar el análisis de la oferta de alojamientos en Madrid.

El objetivo principal del dashboard es facilitar la interpretación de los datos mediante visualizaciones interactivas, permitiendo:

- Filtrar alojamientos por distrito.
- Explorar la distribución de precios y puntuaciones.
- Identificar los alojamientos con más reseñas.
- Analizar la relación entre precio y puntuación.
- Examinar la distribución de estrellas y la presencia de alojamientos de lujo.

5.1 Componentes del Dashboard:

El dashboard cuenta con los siguientes elementos interactivos:

- Selector de distrito: Permite al usuario elegir una zona específica de Madrid para filtrar los alojamientos.
- Mapa interactivo: Visualización geoespacial con la ubicación de los alojamientos, mostrando detalles al pasar el cursor.
- Ranking de distritos: Muestra los distritos mejor valorados según la puntuación media de los alojamientos.
- Gráficos estadísticos:

  ⇒ Histograma de precios: Para analizar la distribución de costos.
  ⇒ Scatter plot (puntuación vs. precio): Para visualizar la relación entre estos dos factores.
  ⇒ Top 10 hoteles con más reseñas: Para identificar los alojamientos más comentados.
  ⇒ Distribución de estrellas: Representación en gráfico de pastel de la proporción de alojamientos según su clasificación.
  ⇒ Ranking de los hoteles más costosos y mejor puntuados.

5.2 Tecnologías Utilizadas

Para el desarrollo del dashboard, se emplearon las siguientes herramientas:

Tecnología	Función
Dash (Python)	Creación del dashboard interactivo.
Plotly	Generación de gráficos dinámicos.
Mapbox	Visualización de datos geoespaciales en el mapa interactivo.

El dashboard fue ejecutado inicialmente en Jupyter Notebook y luego se configuró para su ejecución en un servidor web local, con la posibilidad de ser desplegado en plataformas web.

5.3 Implementación del Dashboard

La construcción del dashboard se llevó a cabo en varias etapas:

a) **Preparación de los Datos:**

Se cargaron los datos previamente limpiados desde el archivo CSV:

```
# Importar librerías necesarias
import dash
from dash import dcc, html  # Importación corregida
from dash.dependencies import Input, Output
import pandas as pd
import plotly.express as px
import random

# Cargar los datos limpios
df = pd.read_csv("alojamientos_booking_limpio.csv")

# Convertir ubicación en coordenadas ficticias (si no tienes
datos reales)
df["Latitud"] = df.index.map(lambda x: 40.40 + random.unifor
m(-0.1, 0.1))
df["Longitud"] = df.index.map(lambda x: -3.70 + random.unifo
rm(-0.1, 0.1))
```

b) **Diseño del Dashboard:** El dashboard fue diseñado con Dash, organizando sus componentes en una interfaz clara e intuitiva:

```
import dash
from dash import dcc, html
from dash.dependencies import Input, Output
# Inicializar la aplicación Dash
app = dash.Dash(__name__)

# Diseño del Dashboard
app.layout = html.Div(children=[
```

```
    html.H1(" Cuadro de Mando - Alojamientos en Madrid", sty
le={'textAlign': 'center'}),

    # Selector de Distrito
    html.Label("🔎 Selecciona un Distrito:"),
    dcc.Dropdown(
        id='distrito-dropdown',
        options=[{'label': 'Todos', 'value': 'Todos'}] +
                [{'label': distrito, 'value': distrito} for d
istrito in df["Distrito"].unique()],
        value='Todos',
        clearable=False
    ),

    # Gráficos dinámicos
    dcc.Graph(id="mapa-hoteles"),
    dcc.Graph(id="ranking-distritos"),
    dcc.Graph(id="hist-precio"),
    dcc.Graph(id="scatter-precio-puntuacion"),
    dcc.Graph(id="top-reseñas"),
    dcc.Graph(id="pie-estrellas"),
    dcc.Graph(id="top-caros"),
    dcc.Graph(id="top-puntuacion"),
    dcc.Graph(id="top-estrellas"),
])

# Callback para actualizar gráficos dinámicamente
@app.callback(
    [Output("mapa-hoteles", "figure"),
     Output("ranking-distritos", "figure"),
     Output("hist-precio", "figure"),
     Output("scatter-precio-puntuacion", "figure"),
     Output("top-reseñas", "figure"),
     Output("pie-estrellas", "figure"),
     Output("top-caros", "figure"),
     Output("top-puntuacion", "figure"),
     Output("top-estrellas", "figure")],
    [Input("distrito-dropdown", "value")]
)
def update_graphs(distrito_seleccionado):
    # Filtrar por distrito
    df_filtrado = df if distrito_seleccionado == 'Todos' els
e df[df["Distrito"] == distrito_seleccionado]
```

5.4 Implementación de los Gráficos Interactivos

El dashboard incorpora diversas visualizaciones, que se actualizan dinámicamente en función del distrito seleccionado.

Mapa Interactivo de Alojamientos

```
    # Filtrar por distrito
    df_filtrado = df if distrito_seleccionado == 'Todos' else df
[df["Distrito"] == distrito_seleccionado]

    # 1 **Mapa Interactivo de Alojamientos**
    fig_mapa = px.scatter_mapbox(
        df_filtrado, lat="Latitud", lon="Longitud", hover_name="
Alojamiento",
        hover_data=["Precio", "Puntuación", "Estrellas", "Reseña
s"],
        color="Puntuación", size="Precio", size_max=15,
        title=" Ubicación de Alojamientos en Madrid",
        color_continuous_scale="viridis",
        zoom=12, height=500
    )
    fig_mapa.update_layout(mapbox_style="open-street-map")
```

Ranking de Distritos por Puntuación

```
   # 2 **Ranking de Distritos por Puntuación Media**
    df_ranking = df.groupby("Distrito")["Puntuación"].mean().res
et_index().sort_values(by="Puntuación", ascending=False)
    fig_ranking_distritos = px.bar(
        df_ranking, x="Puntuación", y="Distrito", orientation='h
',
        title=" Ranking de Distritos con Mejores Puntuaciones",
        color="Puntuación", color_continuous_scale="plasma",
        text_auto=True
    )
```

Histograma de Precios y Scatter Plot - Relación entre Puntuación y Precio

```
    # 3□ **Histograma de Precios**
    fig_hist_precio = px.histogram(
        df_filtrado, x="Precio", nbins=20, title=" Distribución
de Precios",
        color_discrete_sequence=["royalblue"]
    )
    # 4□ **Scatter Plot - Relación entre Puntuación y Precio**
    fig_scatter_precio = px.scatter(
        df_filtrado, x="Puntuación", y="Precio",
        title=" Relación entre Puntuación y Precio",
        color="Puntuación", color_continuous_scale="viridis"
    )
```

Top 10 Alojamientos con Más Reseñas

```
# 5□ **Top 10 Alojamientos con Más Reseñas**
    top_reviews = df_filtrado.nlargest(10, "Reseñas")
    fig_top_reviews = px.bar(
        top_reviews, x="Reseñas", y="Alojamiento", orientation='
h',
        title=" Top 10 Alojamientos con Más Reseñas",
        color="Reseñas", color_continuous_scale="magma",
        text_auto=True
    )
```

Gráfico de Pastel - Distribución de Estrellas

```
# 6□ **Gráfico de Pastel - Distribución de Estrellas**
    fig_pie_estrellas = px.pie(
        df_filtrado, names="Estrellas", title=" Distribución de
Estrellas",
        color_discrete_sequence=px.colors.sequential.Blues
    )
```

Top 10 Hoteles Más Caros

```
# 7□ **Top 10 Hoteles Más Caros**
    top_caros = df_filtrado.nlargest(10, "Precio")
    fig_top_caros = px.bar(
        top_caros, x="Precio", y="Alojamiento", orientation='h',
        title=" Top 10 Hoteles Más Caros",
        color="Precio", color_continuous_scale="reds",
        text_auto=True
    )
```

Top 10 Hoteles con Mejor Puntuación

```
    # 8□ **Top 10 Hoteles con Mejor Puntuación**
    top_puntuacion = df_filtrado.nlargest(10, "Puntuación")
    fig_top_puntuacion = px.bar(
        top_puntuacion, x="Puntuación", y="Alojamiento", orienta
tion='h',
        title=" Top 10 Hoteles con Mejor Puntuación",
        color="Puntuación", color_continuous_scale="teal",
        text_auto=True
    )
```

Top 10 Hoteles con Más Estrellas

```
# 9□ **Top 10 Hoteles con Más Estrellas**
    top_estrellas = df_filtrado.nlargest(min(10, len(df_filtrado
)), "Estrellas")
    fig_top_estrellas = px.bar(
        top_estrellas, x="Estrellas", y="Alojamiento", orientati
on='h',
        title="✨ Top 10 Hoteles con Más Estrellas",
        color="Estrellas", color_continuous_scale="sunsetdark",
        text_auto=True
    )
```

Funcionalidad Dinámica con Callbacks

- Para que los gráficos se actualicen según el distrito seleccionado, se implementó un callback en Dash:

```
@app.callback(
    [Output("mapa-hoteles", "figure"),
     Output("ranking-distritos", "figure"),
     Output("hist-precio", "figure"),
     Output("scatter-precio-puntuacion", "figure"),
     Output("top-reseñas", "figure"),
     Output("pie-estrellas", "figure"),
     Output("top-caros", "figure"),
     Output("top-puntuacion", "figure"),
     Output("top-estrellas", "figure")],
    [Input("distrito-dropdown", "value")]
)

def update_graphs(distrito_seleccionado):
    df_filtrado = df if distrito_seleccionado == 'Todos' else df
[df["Distrito"] == distrito_seleccionado]

    # Se generan los gráficos con los datos filtrados...


    return fig_mapa, fig_ranking_distritos, fig_hist_precio, fig
_scatter_precio, fig_top_reviews, fig_pie_estrellas, fig_top_car
os, fig_top_puntuacion, fig_top_estrellas
```

6. Dashboard Interactivo y Visualización de Datos

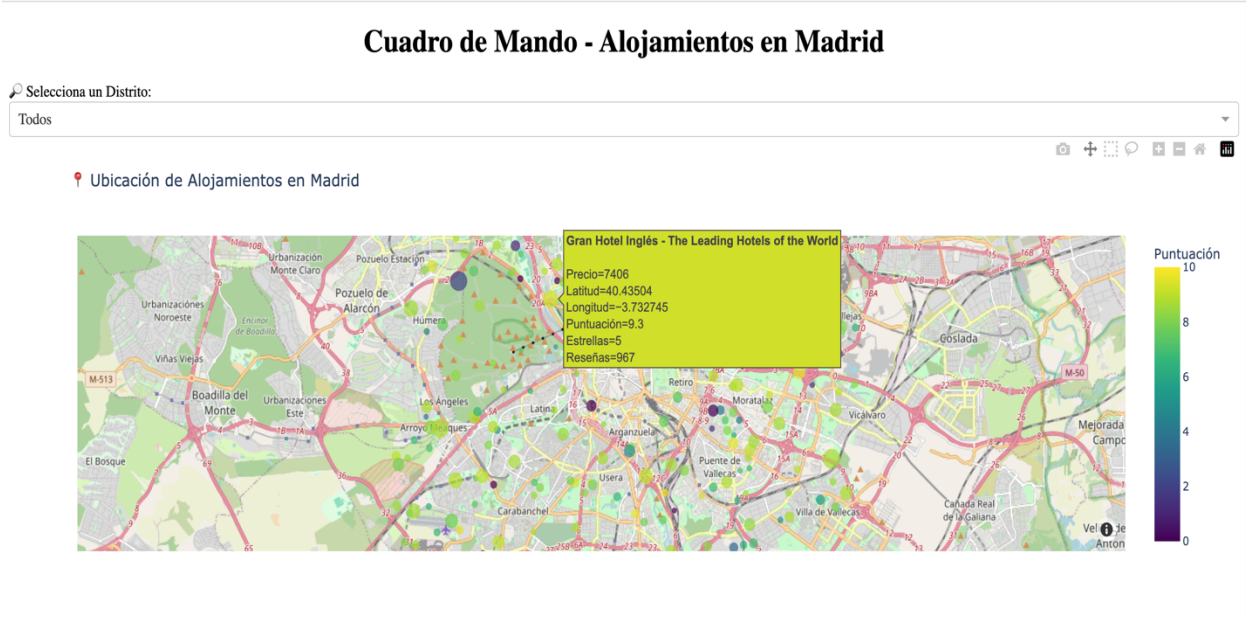
El dashboard cuenta con diversos elementos interactivos que facilitan la exploración de los datos:

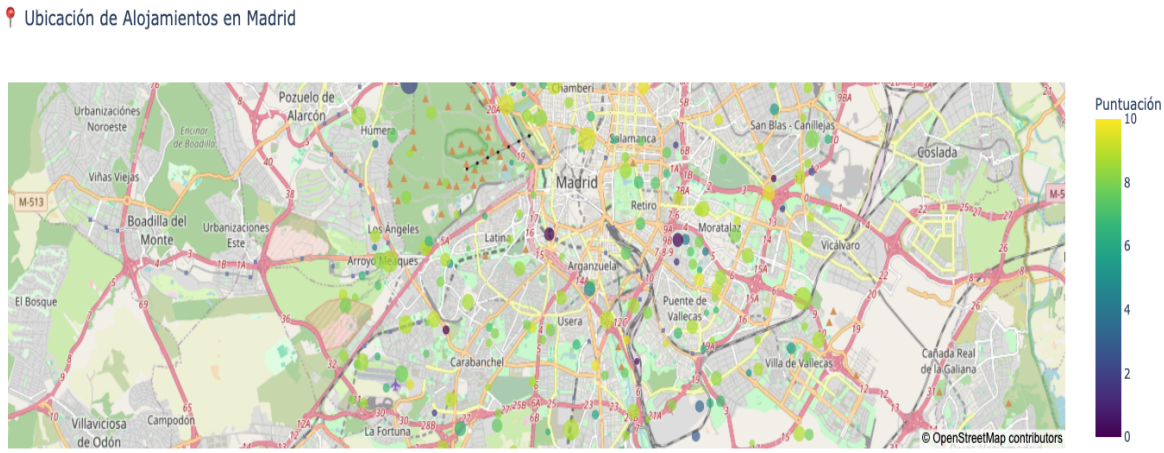
Selector de Distrito

- Menú desplegable que permite al usuario elegir un distrito específico de Madrid para filtrar los alojamientos.
- Todos los gráficos del dashboard se actualizan dinámicamente según la selección.

Mapa Interactivo de Alojamientos

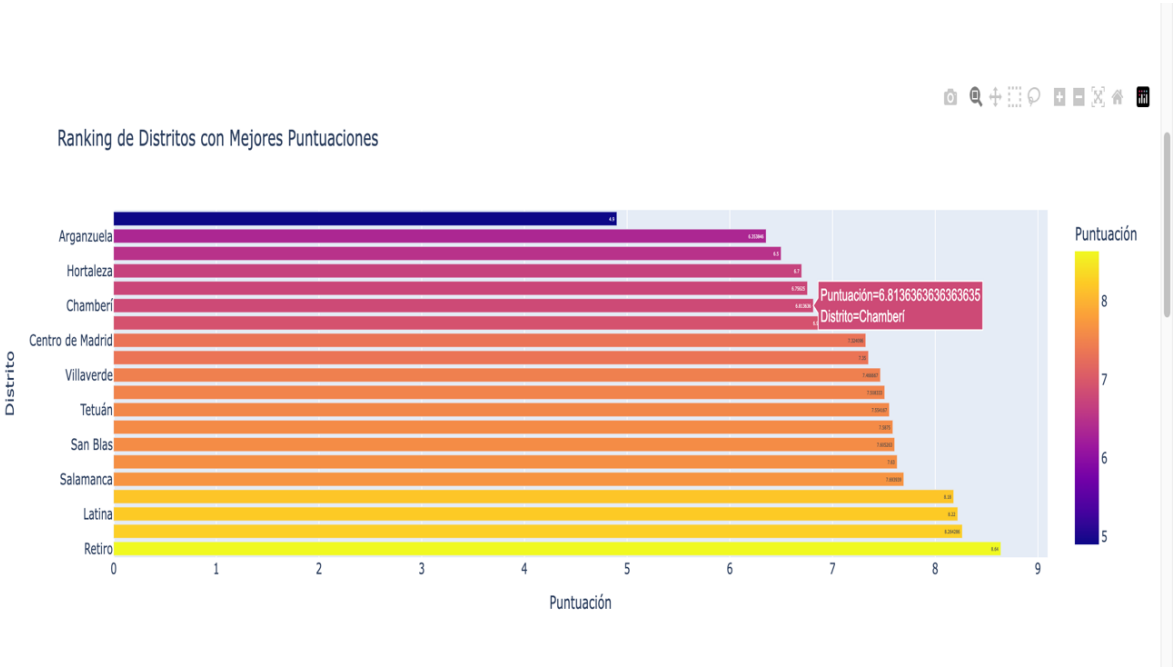
- Se implementó un mapa geoespacial con Mapbox que muestra la ubicación de cada alojamiento.
- Al pasar el cursor sobre un punto en el mapa, se muestra información detallada sobre el alojamiento: nombre, precio, puntuación, número de reseñas y clasificación en estrellas.

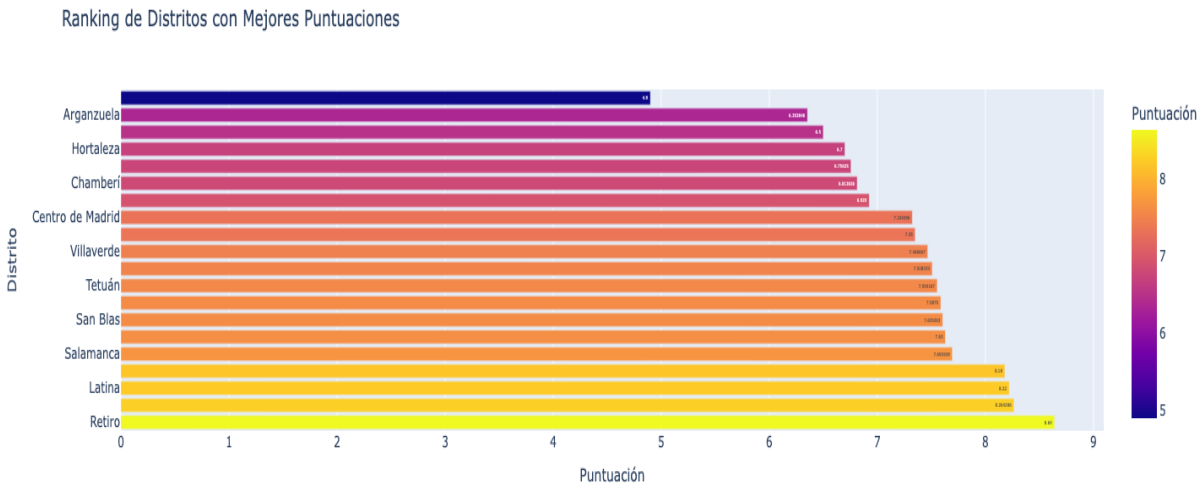




Ranking de Distritos según la Puntuación

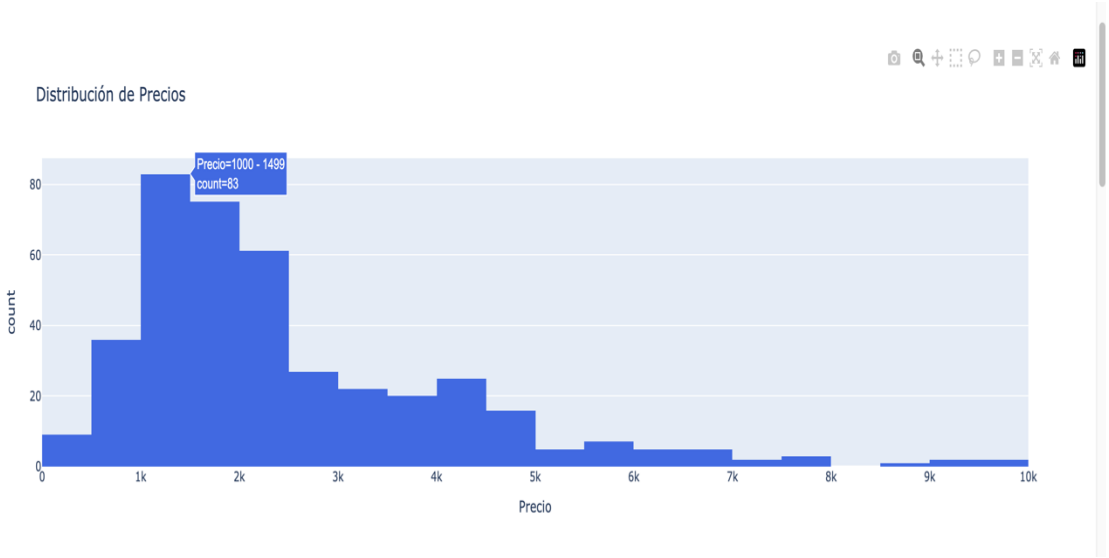
- Un gráfico de barras representa los distritos con mejor puntuación media.
- Se emplea una escala de colores para destacar los distritos con mejores valoraciones.

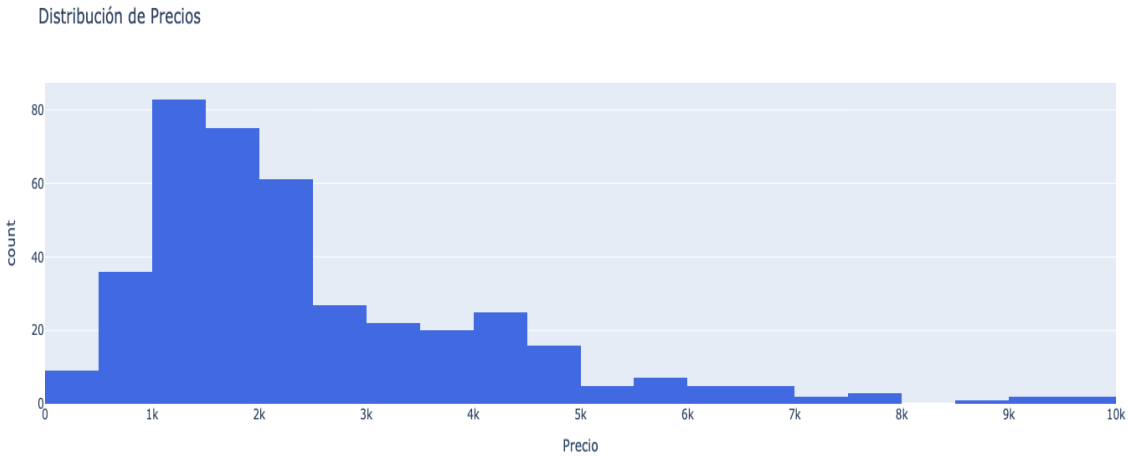




Distribución de Precios

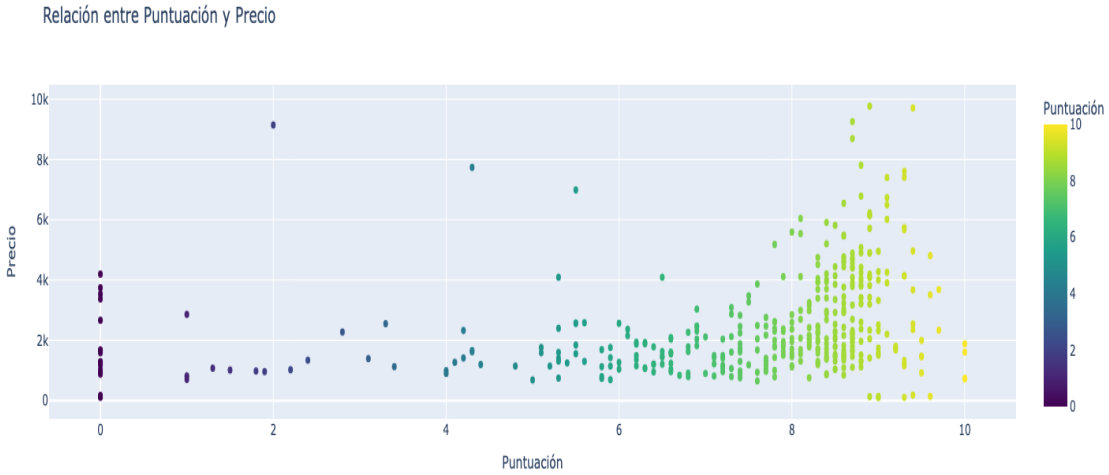
- Se generó un histograma que representa la distribución de los precios de los alojamientos.
- Permite identificar los rangos de precios más comunes y la presencia de valores atípicos (outliers).





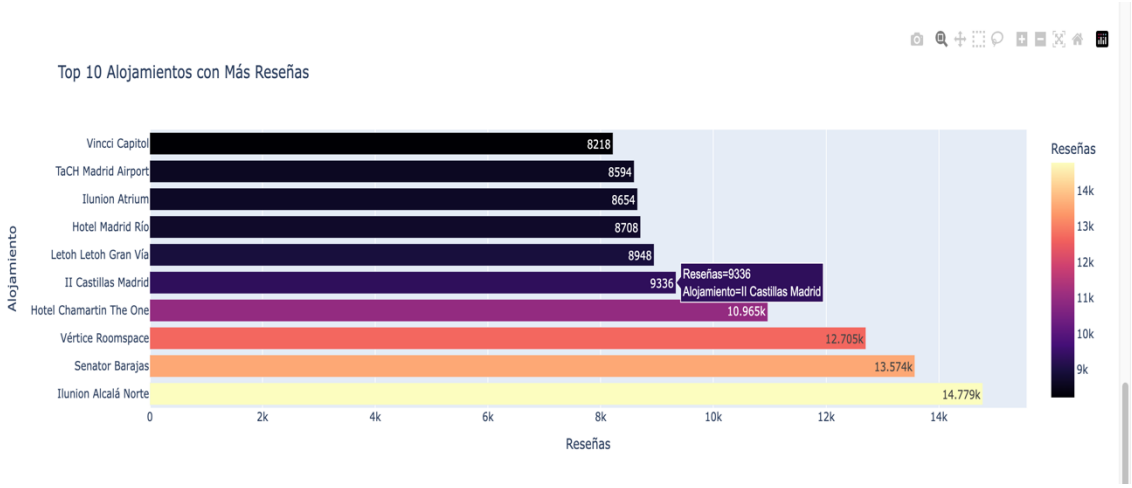
Relación entre Puntuación y Precio

- Se implementó un scatter plot (gráfico de dispersión) para visualizar si existe una correlación entre el precio y la calificación de los alojamientos.

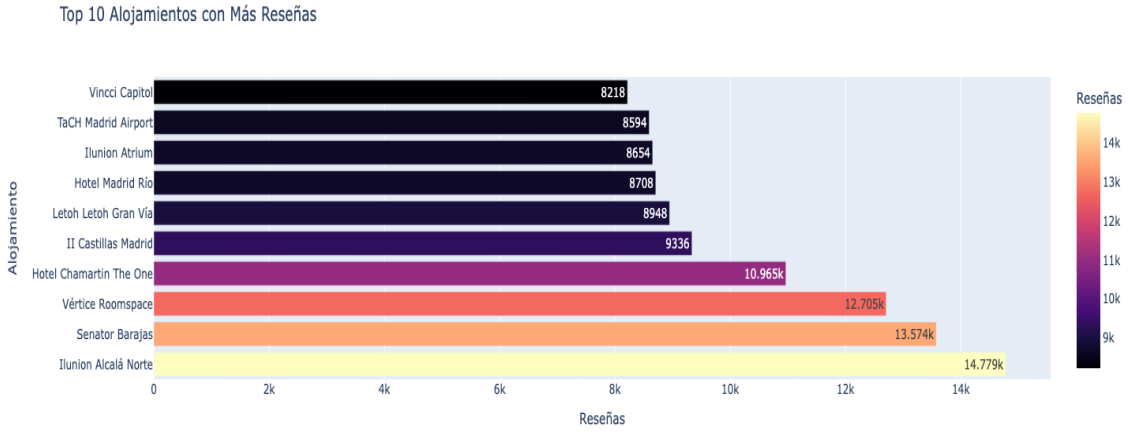


Top 10 Alojamientos con Más Reseñas

- Un gráfico de barras muestra los alojamientos más comentados por los usuarios, lo que permite identificar los establecimientos más populares.

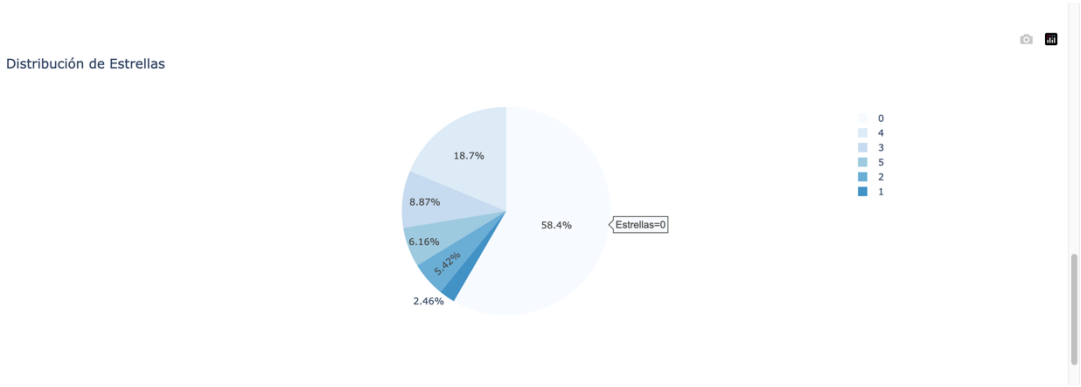


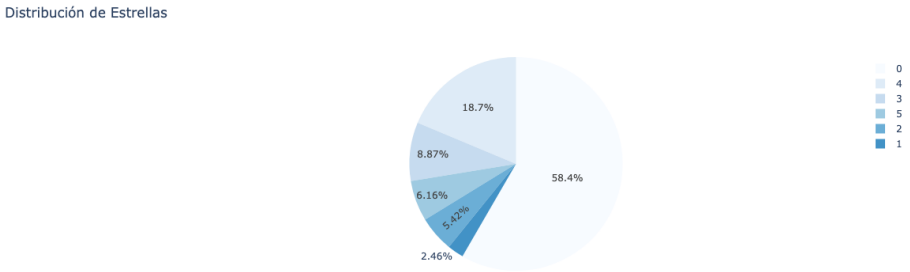
.



Distribución de Estrellas

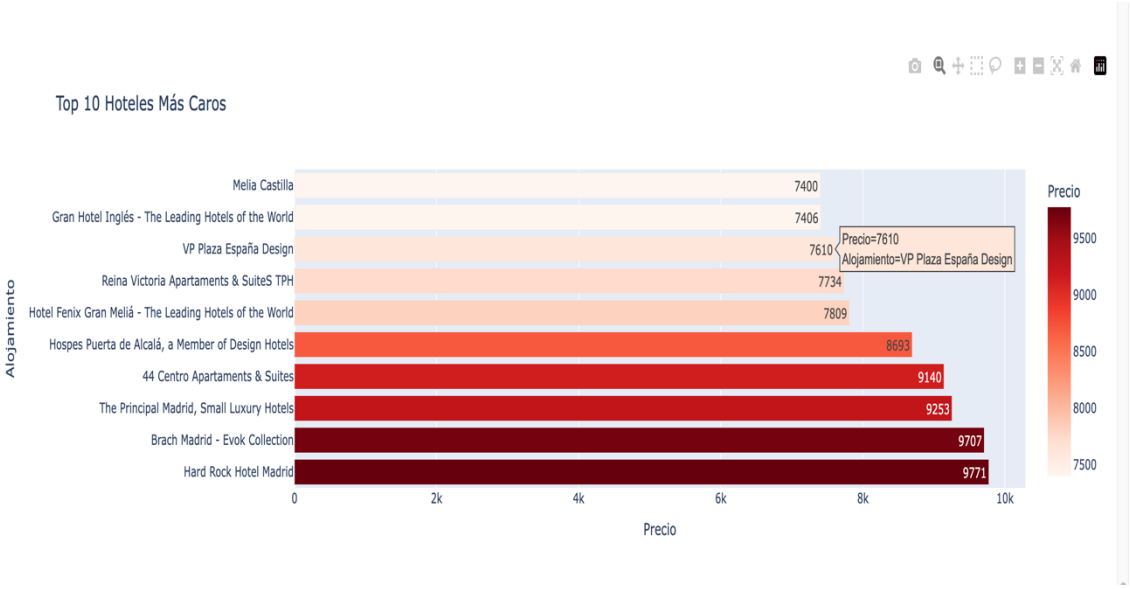
- Un gráfico de pastel representa la proporción de alojamientos según su categoría en estrellas.
- Se puede observar qué porcentaje de alojamientos tiene clasificación oficial y cuántos no poseen estrellas.

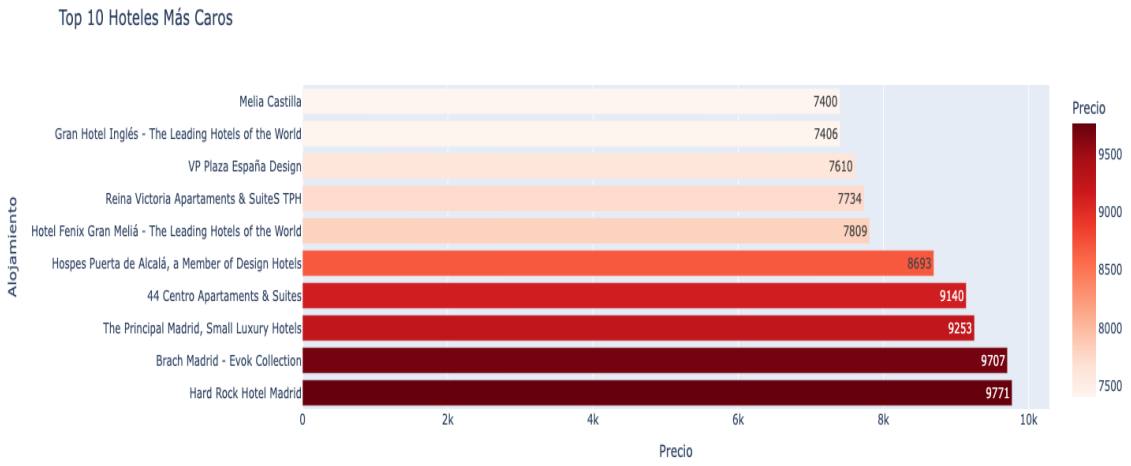




Top 10 Hoteles Más Caros

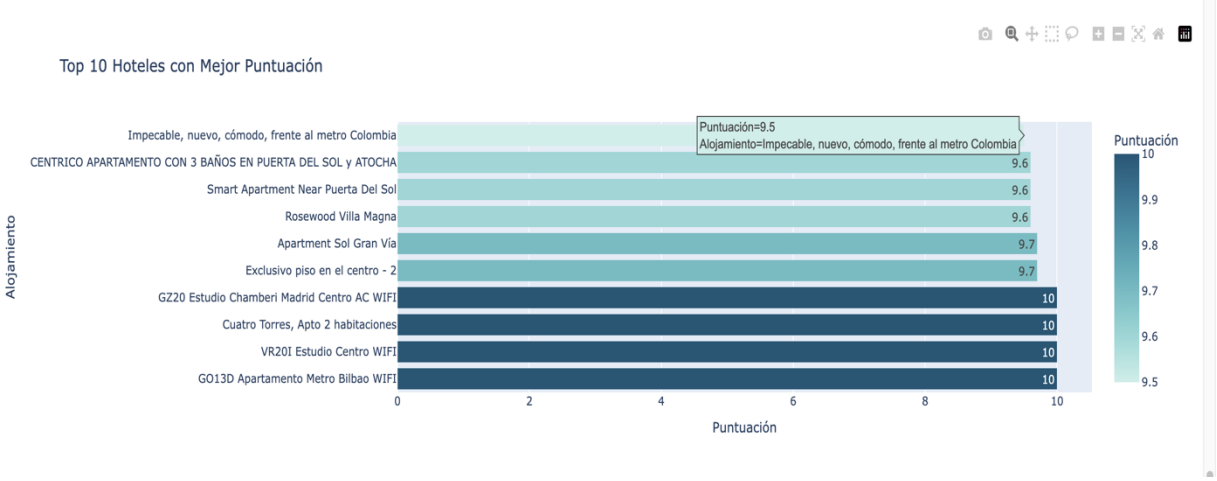
- Un ranking de los alojamientos con los precios más altos de Madrid, facilitando la identificación de los establecimientos más exclusivos.

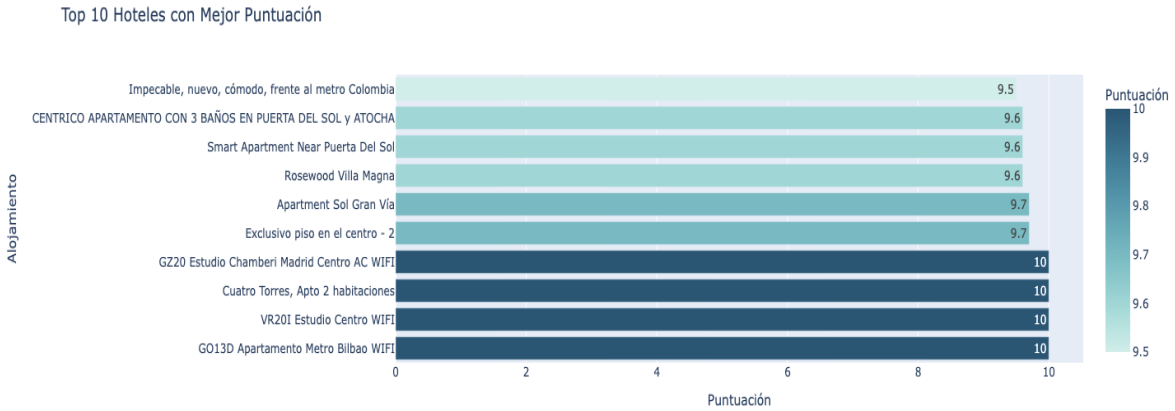




Top 10 Hoteles con Mejor Puntuación

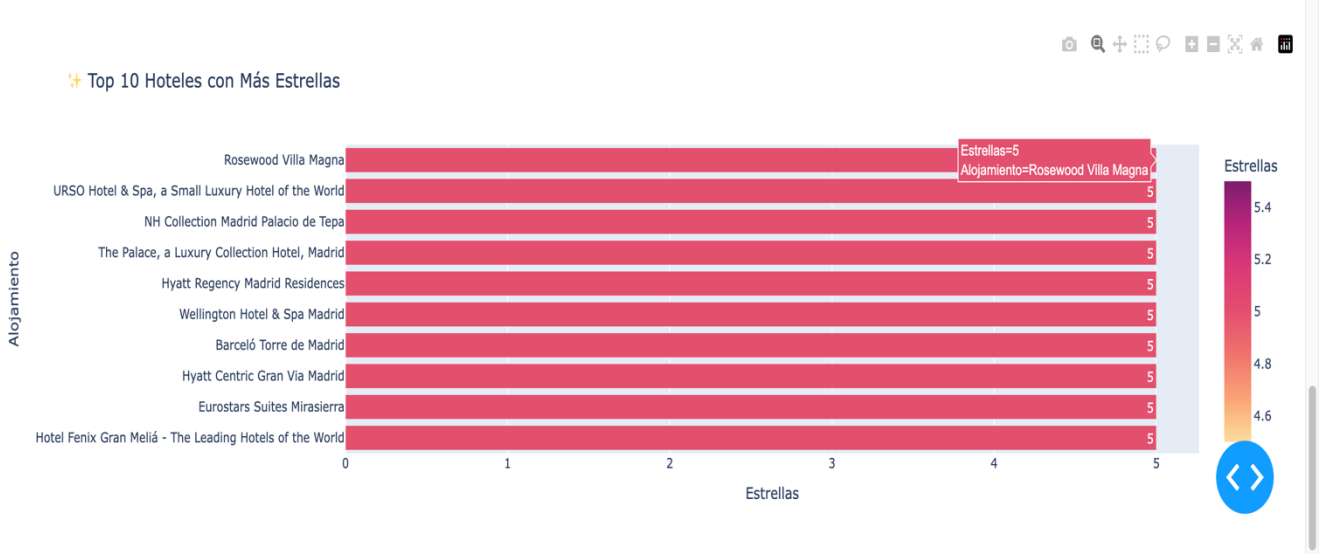
- Se listan los alojamientos con las mejores calificaciones otorgadas por los usuarios.

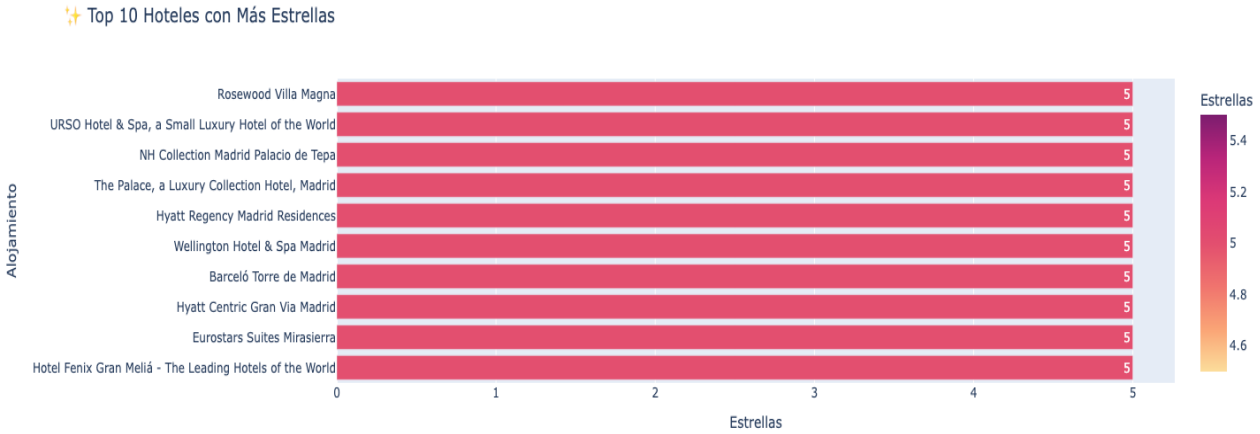




Top 10 Hoteles con Más Estrellas

- Un gráfico de barras destaca los hoteles con la mayor cantidad de estrellas, reflejando la presencia de alojamientos de lujo.





7.Conclusiones

El análisis de los datos de alojamientos en Booking.com para la ciudad de Madrid permitió obtener información valiosa sobre la oferta hotelera, sus precios, puntuaciones y distribución geográfica. Entre los principales hallazgos se destacan:

- ⇒ Distribución de precios: La mayoría de los alojamientos tienen precios entre 1,000€ y 3,000€ por noche, aunque existen opciones de lujo con costos superiores a 9,000€.
- ⇒ Puntuaciones generalmente altas: La mayor parte de los alojamientos tienen valoraciones superiores a 7.0, con un pico en 8.0 y 9.0, lo que indica una percepción positiva por parte de los usuarios.
- ⇒ Relación entre precio y calidad: No se encontró una correlación fuerte entre precio y puntuación. Algunos alojamientos económicos cuentan con altas calificaciones, mientras que algunos costosos no presentan una valoración significativamente mejor.
- ⇒ Hoteles con más reseñas: Un pequeño grupo de alojamientos concentra una gran cantidad de reseñas, con algunos superando las 10,000 opiniones. La mayoría de los alojamientos, sin embargo, tienen menos de 500 comentarios, lo que sugiere menor visibilidad o menor volumen de clientes.
- ⇒ Zonas más populares: Los distritos de Centro de Madrid, Salamanca y Tetuán concentran la mayor cantidad de alojamientos, lo que coincide con su importancia turística y comercial.
- ⇒ Distribución de estrellas: Más del 58% de los alojamientos no posee una clasificación oficial en estrellas, lo que sugiere una amplia oferta de apartamentos turísticos y alojamientos alternativos. Solo un 6.2% de los establecimientos cuenta con 5 estrellas.
- ⇒ Impacto de la cantidad de reseñas en la puntuación: Los alojamientos con mayor número de reseñas tienden a tener una puntuación alta, lo que indica que la confianza de los clientes puede estar influenciada por la cantidad de opiniones disponibles.

Estos hallazgos proporcionan una visión detallada del panorama hotelero en Madrid y pueden ser útiles tanto para viajeros como para empresarios del sector turístico.

El desarrollo del scraping y la visualización en el dashboard presentó varios desafíos técnicos. Booking.com cuenta con protecciones contra automatización, lo que requirió el uso de Selenium con estrategias para evitar bloqueos. Además, la carga dinámica de datos obligó a implementar desplazamientos automáticos y navegación

entre páginas para obtener un conjunto de datos completo. A pesar de estas limitaciones, el dashboard resultante es funcional e interactivo, permitiendo un análisis detallado de la oferta hotelera en Madrid.

El web scraping con Selenium en Booking.com permitió extraer datos clave sobre alojamientos en Madrid, como precios, puntuaciones y reseñas. A pesar de los desafíos técnicos, como la detección de bots y la estructura dinámica del sitio, se logró recopilar y estructurar la información de manera efectiva. Este proceso demostró ser una herramienta poderosa para el análisis de mercado, facilitando la exploración y visualización interactiva de la información. Para futuras mejoras, se podrían explorar el uso de APIs oficiales, optimizar el scraping con técnicas más eficientes o integrar herramientas para mejorar la geolocalización de los alojamientos.

8.Bibliografía

Si usaste artículos, documentación de Python, enlaces web, etc., agrégales una referencia. Durante la realización de este proyecto, se consultaron diversas fuentes para la implementación del scraping, la limpieza de datos y la visualización interactiva. A continuación, se presentan las referencias utilizadas:

Scraping y Automatización Web

⇒ Selenium Documentation: https://www.selenium.dev/documentation/
⇒ BeautifulSoup Documentation: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
⇒ Python Requests Library: https://requests.readthedocs.io/en/latest/

Procesamiento y Limpieza de Datos

⇒ Pandas Documentation: https://pandas.pydata.org/docs/
⇒ NumPy Documentation: https://numpy.org/doc/stable/

Visualización de Datos

⇒ Plotly Python Library: https://plotly.com/python/
⇒ Dash Documentation: https://dash.plotly.com/