Proyecto Júpiter DATA ANALYTICS  
WHITE HOSTING

## MEMORIA Y CRONOGRAMA

línea corta

[MEMORIA Y CRONOGRAMA](#_w8c1ahj5cbz8) 1

[**1. INTRODUCCIÓN**](#_49jzay51708i) **3**

[**2. METODOLOGÍA**](#_qp270t8ytv1h) **4**

[OBJETIVOS DEL ANÁLISIS](#_z7ut5tcmlrp7) 4

[CONJUNTOS DE DATOS UTILIZADOS](#_mf7ik1et7f4q) 4

[TÉCNICAS DE ANÁLISIS Y HERRAMIENTAS UTILIZADAS](#_lc962p9fep99) 6

[PASOS SEGUIDOS PARA EL ANÁLISIS](#_vme0ips7w6g) 7

[Análisis del dataset](#_2ns2insygjkv) 7

[Web scraping Airbnb](#_qqecpsflfpu7) 8

[Unión de csv](#_clbrnad0n3sh) 8

[**4. ANÁLISIS DATASET**](#_ms20iw315kzl) **9**

[LECTURA DE LOS DATOS](#_q55cbkmta2v) 9

[EDA: EXPLORATORY DATA ANALYSIS](#_t17d6ub4kz6x) 9

[COMPRENSIÓN DE LA ESTRUCTURA DE LOS DATOS](#_dueei3fe836z) 9

[BÚSQUEDA DE VALORES NULOS](#_8360s8kjubpf) 9

[REVISIÓN Y ELIMINACIÓN DE DUPLICADOS](#_ogtdv5qvp4ft) 9

[ANÁLISIS ESTADÍSTICO DE LAS VARIABLES](#_tta1nynzy3l) 9

[VALORES ATÍPICOS](#_q6yey72qcqa2) 9

[CORRELACIONES](#_716diqfsrguk) 9

[VARIABLES CREADAS](#_e24qq1hlq2if) 9

[RANKINGS](#_num21rw3gs4v) 9

[HIPÓTESIS](#_yvd13sa2k0zt) 9

[VOLATILIDAD](#_illtpix1livw) 9

[ESTACIONALIDAD](#_1zmdhajxxq5k) 9

[**5. WEB SCRAPING**](#_69d6wf37midi) **9**

[INTRODUCCIÓN](#_prw8zg9u4iry) 9

[IMPORTACIÓN DE LIBRERÍAS](#_v0lkbzo5z89j) 10

[CONFIGURACIÓN DE OPCIONES DE CHROME U CREACIÓN DE WEBDRIVER](#_2xgch04ttvyb) 12

[1. Definir la URL: (ejemplo Madrid)](#_7etzplza8wx4) 12

[2. Conexión:](#_rzy2vyxstu1x) 12

[3. Obtener el HTML de la página:](#_qmoor1lwycxd) 12

[4. Parsear el HTML:](#_15s98adw1h7a) 13

[5. Imprimir información sobre la respuesta HTTP:](#_q3mj2ym7dkzq) 13

[ADQUISICIÓN DE HTML CON ESPERA EXPLÍCITA](#_lx1ks1jysw7e) 14

[EXTRACCIÓN DE ELEMENTOS DE NAVEGACIÓN POR PAGINACIÓN Y CÓDIGO FUENTE](#_5s7y3ad96x3w) 15

[CREAR DF Y GUARDAR EN CSV. FORMATO TEXTO](#_nxsrfbepwiwv) 28

[INFORMACIÓN ADICIONAL](#_i1dox964u5be) 30

[CONCLUSIONES](#_ilz0ar7l8bi5) 30

[**5. PREGUNTAS CLIENTE**](#_2ckcxogbp8ys) **32**

[1. ¿Cuántos inmuebles únicos hay en el dataset vs AirBnb?](#_hp3bytfmrosw) 32

[2. ¿Cuál es la diferencia en número de inmuebles posteados entre las dos fuentes de datos?](#_uoobfn92bh90) 33

[3. ¿Cuáles son los parámetros que conforman el precio?¿Son iguales los parámetros en ambas fuentes de datos?](#_oeitiuyvo0dm) 33

[4. ¿Has podido ver algún inmueble que esté en ambas fuentes de datos?¿Tienen la misma valoración?](#_7j96pyt41jl9) 33

[5. ¿En qué fuente de datos hay más volatilidad en cuanto al precio?¿Hay algún tipo de estacionalidad?](#_pqk58ymdyvq8) 35

[6. Crea un dataset único](#_jcb3hosmc1jt) 36

[EDA para inside\_airbnb 2023](#_5cbbj4p3mosj) 37

[EDA para scrapping\_airbnb 2024](#_d2k12o4y6spr) 38

[Creación del dataset único](#_j0srmvai3u4j) 38

[Conclusiones sobre el Proceso de Unión](#_mvk7m1a0dez1) 38

[Propuesta para Manejar los Desafíos](#_hhx584bewma7) 39

[Conclusión General](#_hpu5j6q4yqe6) 39

[7. ¿Hay alguna variable que podríamos añadir para mejorar el análisis?](#_pfz0ayo3bgyh) 40

[VARIABLE OCUPACIÓN](#_ngx8yj8f62wy) 40

[8. ¿Puedes hacer un ranking de los inmuebles más caros?](#_rnitj65xc33u) 43

[9. ¿Cuáles son los inmuebles más rentables?](#_x2xq0hft73kg) 44

[**6. MODELO ESTRELLA**](#_smnxscc0o3tp) **45**

[TABLA DE HECHOS](#_p3pl1mxta8ci) 45

[TABLAS DE DIMENSIONES](#_owtv4p9kviye) 47

[RELACIONES](#_2l8qmzudn9my) 49

[**7. CONCLUSIONES**](#_9rxa4cwsmgad) **49**

# INTRODUCCIÓN

El principal objetivo del proyecto es proporcionar a White Hosting una visión clara y detallada del mercado inmobiliario español, específicamente en Madrid, Barcelona y Valencia, para respaldar su inversión de 300 millones de euros. Los objetivos específicos incluyen:

* Realizar web scraping de datos de Airbnb para obtener información detallada sobre propiedades vacacionales en las ciudades objetivo.
* Llevar a cabo un análisis exhaustivo de los conjuntos de datos obtenidos, incluyendo la limpieza y análisis preliminar de los mismos.
* Desarrollar hipótesis sobre el mercado inmobiliario español y validarlas mediante análisis de datos.
* Utilizar herramientas de visualización como Tableau o Power BI para presentar los resultados de manera clara y efectiva.
* Responder a preguntas clave planteadas por BlackWidow, como la comparación de datos entre los conjuntos de datos de White Hosting y Airbnb, la identificación de variables relevantes para la inversión y la elaboración de recomendaciones estratégicas.

El proyecto se llevará a cabo utilizando Python como principal lenguaje de programación, haciendo uso de librerías como Pandas para el análisis de datos y Beautifulsoup y Selenium para el web scraping. Además, se utilizarán herramientas de Business Intelligence como Power BI para la presentación visual de los resultados.

En resumen, el proyecto tiene como objetivo proporcionar a White Hosting las herramientas y el análisis necesarios para tomar decisiones informadas sobre su inversión en el mercado inmobiliario español, maximizando su potencial de éxito en un entorno post-pandemia.

# METODOLOGÍA

## OBJETIVOS DEL ANÁLISIS

El objetivo principal del análisis del dataset que nos han proporcionado es identificar patrones y tendencias dentro del mercado inmobiliario a partir de la información de la que disponemos. Aunque el análisis está destinado a las ciudades de Madrid, Barcelona y Valencia, valoraremos información relevante de otras ciudades incluídas en el archivo.

Otro objetivo es poder proporcionar respuestas a las preguntas iniciales que nos presentan en el guión del proyecto. Algunas de ellas podrán obtener respuesta a partir del análisis del dataset, y para otras será necesario el análisis del scrapping que se hará de la web AirBnb.

## CONJUNTOS DE DATOS UTILIZADOS

El conjunto de datos del dataset original está formado por 10.000 registros, con información distribuída en 40 columnas:

* apartment\_id: número identificativo del alojamiento.
* md5: número identificativo del alojamiento..
* name: nombre del alojamiento.
* description: descripción del alojamiento.
* host\_id: número identificativo del anfitrión.
* neighborhood\_overview: descripción del vecindario.
* neighbourhood\_name: nombre del vecindario.
* neighbourhood\_district: distrito del vecindario.
* latitude: latitud.
* longitude: longitud.
* room\_type: tipo de habitación.
* accommodates: número de huéspedes que se pueden alojar en el alojamiento.
* bathrooms: número de baños.
* bedrooms: número de dormitorios.
* beds: número de camas.
* amenities\_list: lista de servicios que ofrece el alojamiento.
* price: precio por noche.
* minimum\_nights: noches mínimas obligatorias.
* maximum\_nights: noches máximas obligatorias.
* has\_availability: disponibilidad.
* availability\_30: disponibilidad en 30 días.
* availability\_60: disponibilidad en 60 días.
* availability\_90: disponibilidad en 90 días.
* availability\_365: disponibilidad en 365 días.
* number\_of\_reviews: número de reseñas.
* first\_review\_date: fecha de la primera reseña.
* last\_review\_date: fecha de la última reseña.
* review\_scores\_rating: calificación de reseñas.
* review\_scores\_accuracy: calificación de la puntuación de exactitud.
* review\_scores\_cleanliness: calificación de la puntuación de limpieza.
* review\_scores\_checkin: puntuación de procedimiento de checkin
* review\_scores\_communication: puntuación de la comunicación.
* review\_scores\_location: puntuación de la localización.
* review\_scores\_value: valor de las puntuaciones.
* license: número de licencia.
* is\_instant\_bookable: reserva inmediata.
* reviews\_per\_month: reseñas al mes.
* country: país.
* city: ciudad.
* insert\_date: fecha en la que el alojamiento se añadió al dataset.

## TÉCNICAS DE ANÁLISIS Y HERRAMIENTAS UTILIZADAS

Para comprender la estructura de los datos del dataset proporcionado, hemos utilizado el Análisis Exploratorio de Datos (EDA) con Python como lenguaje principal. Para realizar este análisis, hemos empleado las librerías:

* Pandas para manipulación y análisis de datos.
* Numpy para operaciones numéricas.
* Seaborn, Plotly Express y Matplotlib para visualización.

Además, para el web scraping de Airbnb y la automatización de tareas, se han empleado:

* Selenium para automatizar la interacción con el navegador web, permitiendo la navegación automatizada por las páginas web objetivo.
* BeautifulSoup para analizar la estructura HTML de la página web y extraer datos específicos de interés de manera eficiente.
* Fake\_usaregent para generar un agente de usuario (User-Agent) aleatorio, lo que puede ayudar a evitar la detección como un robot.
* Request-html para realizar solicitudes HTTP y analizar el contenido HTML de las páginas web.
* Webdriver-manager para ayudar a gestionar y descargar automáticamente el controlador del navegador Chrome.
* Upgrade webdriver-manager, para actualizar la última versión del anterior.

Para la instalación de estas librerías se ha usado el comando “pip instal”. Como se muestran a continuación:

!pip install selenium

!pip install fake\_useragent

!pip install requests-html

!pip install webdriver-manager

!pip install --upgrade webdriver-manager

## PASOS SEGUIDOS PARA EL ANÁLISIS

#### Análisis del dataset

1. Importación de librerías.
2. Se procedió a la carga de datos proporcionados en formato csv
3. Análisis Exploratorio de Datos (EDA)
   1. Comprensión de los datos: Se realizó una exploración inicial de los datos para comprender su estructura y contenido. Esto incluyó revisar algunas filas del DF, así cómo comprobar su tamaño y tipos de datos por columnas.
   2. Búsqueda de valores nulos: se identificaron y gestionaron los valores nulos en el dataset.
   3. Revisión de duplicados. Se verificó la presencia de filas duplicadas en el dataset y se tomaron medidas para manejarlos según las necesidades.
   4. Análisis Estadístico de las variables significativas: Se realizó un análisis estadístico descriptivo de las variables más relevantes del dataset, calculando medidas como valores máximos y mínimos, etc.
   5. Valores atípicos: se detectaron valores atípicos en el dataset mediante técnicas de visualización y se procedió a su eliminación.
   6. Correlaciones: se exploraron las relaciones entre las diferentes variables mediante el cálculo de coeficientes de relación.
4. Creación de variables y rankings de inmuebles.
   1. Se procedió a crear nuevas variables derivadas de los datos existentes en el dataset, con el fin de obtener información adicional relevante para el análisis.
   2. Se desarrollaron una serie de rankings clasificatorios basados en ciertas características de interés para el proyecto.
5. Se elaboraron hipótesis y contestaron a las preguntas del cliente.
6. Guardado del data frame limpio en csv.

#### Web scraping Airbnb

1. Importación de librerías.
2. Configuración de opciones de Chrome y Creación de WebDriver:
   1. Definir la URL
   2. Conexión y espera
   3. Obtención del HTML
   4. Parseo del HTML con BeautifulSoup
   5. Impresión de Información de la página.
3. Adquisición de HTML con Espera explícita
4. Extracción de elementos y navegación por páginas y código fuente.
5. Creación de df y guardado en csv.

#### Unión de csv

Como paso final del análisis, se unificaron los csv resultantes tanto del análisis del dataset inicial, como del web scraping de Airbnb.

# 4. ANÁLISIS DATASET

## LECTURA DE LOS DATOS

Se realizó la lectura de los datos provenientes del dataset proporcionado por White Hosting y para el estudio del mercado de alojamientos vacacionales. Contiene datos hasta el año 2021.

## EDA: EXPLORATORY DATA ANALYSIS

Se llevó a cabo un análisis exploratorio de datos para entender la naturaleza y características de los datos. Esto incluyó la visualización de variables, distribución de datos, identificación de tendencias y patrones iniciales.

### COMPRENSIÓN DE LA ESTRUCTURA DE LOS DATOS

Se profundizó en la estructura de los datos para comprender la relación entre las diferentes variables, identificar variables clave y entender la lógica subyacente de los datos.

### BÚSQUEDA DE VALORES NULOS

Se realizó una búsqueda exhaustiva de valores nulos o faltantes en el conjunto de datos para determinar su impacto en el análisis y tomar decisiones sobre cómo manejar estos valores ausentes.

Decidimos no trabajar con aquellas columnas que presentaban gran cantidad de nulos, tales como:

license 4856

neighborhood\_overview 4101

neighbourhood\_district 3921

review\_scores\_value 2724

review\_scores\_location 2724

review\_scores\_checkin 2723

review\_scores\_accuracy 2718

review\_scores\_communication 2714

review\_scores\_cleanliness 2712

review\_scores\_rating 2709

last\_review\_date 2605

first\_review\_date 2604

reviews\_per\_month 2604

has\_availability 550

La variable price, poseía 254 nulos, que también fueron eliminados porque no nos iban a servir para nuestro análisis.

### REVISIÓN Y ELIMINACIÓN DE DUPLICADOS

Se revisaron los datos en busca de registros duplicados y se tomaron medidas para eliminarlos, asegurando la integridad y precisión de los datos.

Comprobamos que había valores duplicados, es decir, que se habían cargado más de una vez en el dataset. Decidimos quedarnos con el valor que tuviera la fecha de reseña más reciente, y eliminar los demás

### ANÁLISIS ESTADÍSTICO DE LAS VARIABLES

PRICE

Comprobamos que, del total de alojamientos, había 8639 alojamientos con un precio inferior a 200 euros 997 alojamientos con un precio superior a 200. Las ciudades donde se concentran dichos alojamientos con mayores precios son: Mallorca, Barcelona, Girona y Madrid. Obviamente el alojamiento con el mayor precio es el 'apartamento entero'. Podemos deducir en cuanto al precio que el alojamiento más interesante en cuanto a rentabilidad es el apartamento entero y las ciudades que tienen mayores precios son: Mallorca, Barcelona, Girona y Madrid.

CITY

Donde más alojamientos hay es en las ciudades de Barcelona, Madrid, Mallorca y Girona.

En cuanto a rentabilidad, en términos de precio, Mallorca, Menorca y Girona son las ciudades más caras. Sin embargo, Madrid y Barcelona tienen una media de estancia mayor en número de días de hospedaje.

ACCOMMODATES

Mallorca, Barcelona, Madrid y Girona son las ubicaciones con la mayor capacidad de huéspedes, principalmente debido a que la mayoría de los alojamientos son apartamentos enteros. Resulta notable que las dos ciudades más grandes del país presentan un número significativo de "habitaciones privadas". Esta tendencia podría atribuirse al considerable número de alojamientos de este tipo dispersos en ambas ciudades, dado su estatus como las más pobladas del país.

AVAILABILITY\_365

Es evidente que Madrid y Barcelona cuenten con una mayor disponibilidad de alojamientos turísticos a lo largo del año, dado que son ciudades que se visitan frecuentemente durante los fines de semana.

En contraste, las ciudades costeras y las islas presentan una disponibilidad más reducida, ya que suelen ser destinos más populares en verano o en condiciones climáticas favorables. Otra explicación plausible sería la presencia de alquileres temporales destinados a otros segmentos de la población, como estudiantes o profesores.

ROOM\_TYPE

El tipo de alojamiento más demandado y con mayor concentración en todas las ciudades del DF, es el apartamento entero. También es obvio que es el que más rentabilidad da dado el precio medio.

BATHROOMS

Como era de esperar, el alojamiento entero es el que más número de baños tiene al concentrar más del 50% de la oferta

BEDROOMS

La mayoría de alojamientos del dataset tienen entre 1 y 2 habitaciones. Más de 6000 habitaciones, pertenecen a alojamientos enteros, 68 a habitaciones de hotel, casi 3000 son habitaciones privadas dentro de alojamientos y 69 habitaciones compartidas. La media total es de casi 2 habitaciones por alojamiento.

### VALORES ATÍPICOS

Los valores atípicos que nos llamaron más la atención fueron de las variables:

* Baños. Había dos alojamientos en Mallorca con un número elevado de baños. Pero comprobamos que eran Villas. Por lo tanto, tenía sentido.
* Huéspedes.
* Habitaciones. Un alojamiento mostraba 50 habitaciones en Madrid.
* Noches mínimas. Un alojamiento mostraba un mínimo de más de 1000 noches, en Madrid. Lo cual no era coherente.

Se establecieron umbrales para eliminar los outliers.

### CORRELACIONES

El análisis de la correlación nos mostró que las variables que tenían una relación directa con el precio eran el número de huéspedes, las habitaciones y las camas. Comprobamos que el valor de las reseñas no tenía nada que ver con el precio.

### VARIABLES CREADAS

\*OCUPACIÓN

\*RENTABILIDAD

\*ESTACIONALIDAD

\*Están explicadas en el punto PREGUNTAS DEL CLIENTE.

### RANKINGS

\*\*RANKINGS DE INMUEBLES MÁS CAROS

\*\*RANKINGS INMUEBLES MÁS CAROS POR RANGO DE PRECIOS

\*\*RANKINGS TENIENDO EN CUENTA LA OCUPACIÓN Y LA RENTABILIDAD

\*\*Están explicadas en el punto PREGUNTAS DEL CLIENTE.

### HIPÓTESIS

## \*\*\*VOLATILIDAD

## \*\*\*ESTACIONALIDAD

\*\*\* Están explicadas en el punto PREGUNTAS DEL CLIENTE.

# 5. WEB SCRAPING

## INTRODUCCIÓN

El propósito de esta sección es detallar meticulosamente el proceso seguido para el web scraping en Airbnb, desde la preparación técnica inicial con la importación de librerías y la configuración del entorno de navegación automatizada con Selenium, hasta la extracción y procesamiento de datos específicos de interés. Este enfoque nos permite generar un conjunto de datos complementario al dataset original, enriqueciendo significativamente nuestro análisis y ofreciendo una base de evidencia robusta para respaldar las decisiones estratégicas de inversión de White Hosting.

## IMPORTACIÓN DE LIBRERÍAS

Para llevar a cabo el proceso de web scraping en este proyecto, hacemos uso de una serie de librerías de Python especializadas en la automatización web, el análisis de datos y la gestión de HTML. Estas herramientas son fundamentales para navegar por la web, extraer información relevante y estructurar los datos de manera que puedan ser analizados eficazmente. A continuación, se detalla la funcionalidad y el propósito detrás de cada librería empleada:

* Selenium: Esencial para automatizar la interacción con páginas web a través de un navegador. Nos permite simular acciones del usuario como clics y navegación, crucial para acceder a datos dinámicos que se cargan mediante JavaScript.
* BeautifulSoup: Utilizada para parsear documentos HTML y XML. Facilita la extracción de datos de páginas web, permitiéndonos acceder a contenido específico y estructurarlo adecuadamente para su análisis.
* Pandas: Brinda estructuras de datos potentes y flexibles, como DataFrames, que facilitan la manipulación y el análisis de grandes conjuntos de datos. Es instrumental para organizar los datos extraídos en una forma que sea fácilmente analizable.

Para instalar estas librerías en el entorno de desarrollo, se utilizan los siguientes comandos:

!pip install selenium

!pip install beautifulsoup4

!pip install pandas

Además, se emplean librerías complementarias para mejorar la eficiencia del scraping y manejar detalles específicos del navegador y las solicitudes HTTP:

* fake\_useragent: Genera agentes de usuario aleatorios para simular accesos al sitio web desde diferentes navegadores y sistemas operativos, ayudando a evitar bloqueos por parte de los servidores web.
* webdriver-manager: Simplifica el manejo de los controladores de navegador necesarios para Selenium, automatizando su descarga y actualización.
* requests-html: Aunque en este contexto primamos el uso de Selenium, requests-html puede ser útil para realizar solicitudes HTTP directas a páginas web y analizar contenido estático.

Los comandos para instalar estas librerías adicionales son:

!pip install fake\_useragent

!pip install webdriver-manager

!pip install requests-html

Finalmente, se realiza la importación de las librerías y módulos necesarios en el código fuente, estableciendo la base para el desarrollo del proceso de web scraping y análisis de datos:

import time

from bs4 import BeautifulSoup

from selenium import webdriver

from selenium.webdriver.chrome.options import Options

from fake\_useragent import UserAgent

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

from selenium.webdriver.chrome.service import Service

from webdriver\_manager.chrome import ChromeDriverManager

import pandas as pd

from datetime import datetime

Cada una de estas importaciones tiene un propósito específico, desde gestionar el tiempo de espera entre solicitudes hasta procesar y estructurar los datos extraídos. Juntas, forman el conjunto de herramientas que nos permitirán ejecutar el web scraping de forma eficiente y respetuosa, maximizando la calidad de los datos recopilados para White Hosting.

## CONFIGURACIÓN DE OPCIONES DE CHROME U CREACIÓN DE WEBDRIVER

La configuración de las opciones de Chrome y la creación de un objeto WebDriver son pasos fundamentales para iniciar la automatización del navegador en tareas de web scraping. Esta etapa permite ajustar las preferencias de navegación para optimizar el rendimiento del scraping y asegurar una extracción de datos efectiva y eficiente. A continuación, se describe el proceso detallado para White Hosting, con el objetivo de maximizar la recopilación de datos sobre alojamientos vacacionales.

### Definir la URL: (ejemplo Madrid)

airbnb\_url\_madrid = f'https://www.airbnb.es/s/Madrid--España/homes?'

En esta línea, se define una variable llamada airbnb\_url\_madrid que contiene la URL de la página de Airbnb que se desea analizar. La URL incluye la ubicación (Madrid, España) y la categoría de alojamiento (en este caso, "homes").

### Conexión:

driver.get(airbnb\_url\_madrid)

time.sleep(10) # Esperar a que se carguen los datos

Aquí, se utiliza Selenium para abrir la URL en un navegador Chrome controlado por el script. driver.get(airbnb\_url\_madrid) abre la URL en el navegador. Luego, time.sleep(10) pausa la ejecución del script durante 10 segundos, permitiendo que la página web se cargue completamente antes de continuar.

### Obtener el HTML de la página:

html = driver.page\_source

Esta línea obtiene el contenido HTML de la página web cargada utilizando Selenium y lo almacena en la variable html.

### Parsear el HTML:

soup = BeautifulSoup(html, 'lxml') (con lxml se supone que va más rápido)

Aquí, se utiliza BeautifulSoup, una biblioteca de Python para analizar documentos HTML y XML, para parsear el contenido de la respuesta HTTP (que es el HTML de la página web). Se crea un objeto BeautifulSoup llamado soup que facilita la extracción de datos de la página web.

### Imprimir información sobre la respuesta HTTP:

print(driver.current\_url) # URL final después de las redirecciones

print(driver.title) # Título de la página

1. Estas líneas imprimen información sobre la respuesta HTTP obtenida:
   * driver.current\_url muestra la URL final después de que se hayan seguido posibles redirecciones.
   * driver.title muestra el título de la página web actual.

En resumen, este código realiza una serie de acciones para abrir la página de Airbnb de Madrid en un navegador controlado por Selenium, obtiene el contenido HTML de la página y luego utiliza BeautifulSoup para facilitar la extracción de datos de esa página. Las líneas finales imprimen información básica sobre la página web actual. Este código sería el punto de partida para realizar operaciones adicionales de extracción de datos o análisis en la página web de Airbnb.

## ADQUISICIÓN DE HTML CON ESPERA EXPLÍCITA

El documento detalla métodos para cargar eficientemente la página web airbnb y extraer contenido HTML. Se enfatiza el uso de esperas explícitas para asegurar que los elementos web estén completamente cargados antes de la extracción de datos.

1. airbnb\_url\_madrid = f'https://www.airbnb.es/s/Madrid--España/homes?': Se define la URL de la página web de Airbnb que se desea analizar. Esta URL contiene la búsqueda de alojamientos en Madrid, España.
2. driver.get(airbnb\_url\_madrid): Se utiliza Selenium para abrir la URL en un navegador Chrome controlado por el script. Esto carga la página web en el navegador.
3. time.sleep(10): Se añade una pausa de 10 segundos para permitir que la página web se cargue completamente. Esto es útil para asegurarse de que todos los elementos de la página se hayan cargado antes de continuar.
4. html = driver.page\_source: Se utiliza driver.page\_source para obtener el HTML de la página web actualmente cargada en el navegador controlado por Selenium. Esto captura todo el contenido HTML de la página en una variable llamada html.
5. soup = BeautifulSoup(html, 'lxml'): Se utiliza BeautifulSoup para parsear el contenido HTML almacenado en la variable html. La cadena 'lxml' es el analizador que BeautifulSoup utilizará para analizar el HTML. BeautifulSoup crea un objeto soup que facilita la extracción de datos específicos del HTML.
6. print(driver.current\_url): Se imprime la URL final después de posibles redirecciones. Esto es útil para verificar si la URL ha cambiado después de seguir enlaces o redireccionamientos.
7. print(driver.title): Se imprime el título de la página web actualmente cargada en el navegador controlado por Selenium. Esto proporciona información sobre el título de la página, que puede ser útil para identificar la página web.

## EXTRACCIÓN DE ELEMENTOS DE NAVEGACIÓN POR PAGINACIÓN Y CÓDIGO FUENTE

Se discuten técnicas para extraer elementos específicos de las páginas web y navegar a través de resultados paginados. Los ejemplos incluyen la extracción de detalles de listados de anuncios por cada ciudad y la navegación a través las páginas para obtener las características individuales de cada anuncio.

Y además, se explica con detalle el código fuente de Python utilizado en el proyecto. Incluye comentarios para mayor claridad y explica la funcionalidad de cada bloque de código.

1. **Acceso a Google Drive**: Guardar automáticamente los csv en el Google *\*Drive\**
2. **En general: En todos los botones se deja esto, porque a veces da fallo y hay que volver a ejecutar el drive.**

# import time # Librería para controlar el tiempo

# from selenium import webdriver # Librería para controlar el navegador

# from selenium.webdriver.chrome.options import Options # Librería para controlar el navegador

# from selenium.webdriver.common.by import By # Librería para controlar el navegador

# from selenium.webdriver.support.ui import WebDriverWait # Librería para controlar el navegador

# from selenium.webdriver.support import expected\_conditions as EC # Librería para controlar el navegador

# from fake\_useragent import UserAgent # Librería para controlar el navegador

# import pandas as pd # Librería para controlar el navegador

# # Configuración inicial del WebDriver y opciones

# ua = UserAgent() # Crear un objeto UserAgent

# chrome\_options = Options() # Crear un objeto Options

# chrome\_options.add\_argument('--headless') # Modo headless

# chrome\_options.add\_argument('--no-sandbox')

# chrome\_options.add\_argument('--disable-dev-shm-usage')

# chrome\_options.add\_argument("--disable-extensions")

# chrome\_options.add\_argument(f'user-agent={ua.random}')

# chrome\_options.headless = True #

# driver = webdriver.Chrome(options=chrome\_options) # Crear un objeto WebDriver

**Botón 1 :**  Botón 1 es la primera página no hace falta botón clic

Explicación detallada de cada paso.

1. anuncios\_pisos\_pag1 = []: Se crea una lista vacía llamada anuncios\_pisos\_pag1 que se utilizará para almacenar la información de cada anuncio en la página actual.
2. time.sleep(10): Se hace una pausa de 10 segundos para esperar a que la página cargue. Esto es útil después de hacer clic en el botón de siguiente página para asegurarse de que la página siguiente se haya cargado completamente.
3. try:: Se inicia un bloque de código que maneja posibles excepciones.
4. elementos\_anunciospag1 = WebDriverWait(driver, 10).until(...): Se utiliza WebDriverWait para esperar hasta que todos los elementos de anuncios con la clase 'c4mnd7m' estén presentes en el DOM dentro de un límite de tiempo de 10 segundos. Estos elementos son los anuncios en la página actual.
5. Se realizan operaciones similares de espera para otros elementos, como precios, subtítulos, valoraciones, títulos y precios totales.
6. for index, anunciopag1 in enumerate(elementos\_anunciospag1):: Se inicia un bucle for que recorre todos los elementos de anuncios en la página actual. enumerate se utiliza para obtener tanto el índice (index) como el elemento actual (anunciopag1) en cada iteración.
7. Se extraen datos específicos de cada anuncio, como el título, subtítulo, valoración, precio por noche, precio total, enlace (href) y objetivo (target) del enlace. Si alguno de estos datos no está presente, se establece como 'N/A' para manejar casos en los que falta información.
8. Se imprime la información del anuncio actual para fines de depuración y se utiliza "-" \* 50 para imprimir una línea divisoria.
9. Se agrega un diccionario con la información del anuncio actual a la lista anuncios\_pisos\_pag1 utilizando append.
10. Se manejan posibles excepciones internas en el bucle for para asegurarse de que cualquier error no detenga todo el proceso de extracción de datos.
11. Después de recopilar la información de todos los anuncios en la página actual, se cierra el bloque try.
12. except Exception as e:: Si ocurre alguna excepción durante la extracción de datos de los anuncios, se captura y se imprime un mensaje de error.
13. if anuncios\_pisos\_pag1:: Se verifica si se obtuvieron enlaces de anuncios en la página actual.
14. for anuncio in anuncios\_pisos\_pag1:: Si se obtuvieron enlaces de anuncios, se inicia un bucle for para recorrer estos enlaces y realizar scraping de características específicas de cada anuncio.
15. WebDriverWait(driver, 10).until(...): Se espera a que las características estén presentes en el DOM utilizando WebDriverWait. Estas características se identifican por su clase "lgx66tx.atm\_gi\_idpfg4.atm\_l8\_idpfg4.dir.dir-ltr".
16. Se obtienen las características y se almacenan en caracteristicas\_texto.
17. Se imprime opcionalmente el texto de las características.
18. Se agrega el texto de las características al diccionario del anuncio actual bajo la clave 'caracteristicas'.
19. Se manejan posibles excepciones internas para asegurarse de que cualquier error durante el scraping de características no detenga todo el proceso.
20. except Exception as e:: Si ocurre alguna excepción durante el scraping de características, se captura y se imprime un mensaje de error.
21. Si no se obtuvieron enlaces de anuncios en la página actual, se imprime un mensaje indicando que no se realizará el scraping.

**Botón 2 y 3 :**  Botón 2 y 3 se hace con driver.find\_element\_by\_xpath ya que están visibles

1. boton\_2 = driver.find\_element(By.XPATH, "//a[contains(@class, 'atm\_1y33qqm\_1ggndnn') and contains(text(), '2')]"): Esta línea de código busca un elemento en la página web utilizando una expresión XPath. El elemento que se está buscando es un enlace <a> que contiene la clase "atm\_1y33qqm\_1ggndnn" y el texto "2". En otras palabras, busca el enlace que lleva a la página 2.
2. A continuación, el código intenta hacer clic en el enlace encontrado en el paso anterior utilizando boton\_2.click(). Esto se hace para navegar a la página 2 de resultados en Airbnb.
3. time.sleep(5): Después de hacer clic en el enlace, el código espera 5 segundos. Esto se hace para dar tiempo a que la página 2 se cargue completamente antes de continuar.
4. print("URL actual después de hacer clic en el botón 2:", driver.current\_url): Imprime la URL actual después de hacer clic en el botón 2. Esto es útil para verificar que la navegación a la página 2 haya sido exitosa y para obtener la URL actual.
5. time.sleep(10): El código espera 10 segundos antes de continuar. Esto se hace para asegurarse de que la página 4 (o la página siguiente después de hacer clic en el botón 2) se cargue completamente.
6. Luego, el código inicia una nueva lista llamada anuncios\_pisos\_pag2 para almacenar la información de cada anuncio en la página 2.
7. El código utiliza Selenium y WebDriverWait para esperar hasta que los elementos de los anuncios estén presentes en el DOM de la página 2. Esto se hace para asegurarse de que los anuncios se hayan cargado antes de intentar extraer información de ellos.
8. Se repiten los mismos pasos que en el código anterior para extraer información de los anuncios, como el título, el subtítulo, la valoración, el precio por noche, el precio total, el enlace y el target del anuncio.
9. También se maneja la posibilidad de que no se pueda encontrar alguno de los elementos y se asigna 'N/A' en caso de que falten datos.
10. El código intenta realizar scraping de las características específicas de cada anuncio utilizando un bucle for. Se hace clic en el enlace de cada anuncio, se espera a que las características estén presentes en el DOM y se extraen.
11. Las características se almacenan en una lista llamada caracteristicas\_texto.
12. Se imprime el texto de las características, y las características se añaden al diccionario del anuncio correspondiente.
13. Finalmente, el código intenta guardar los resultados en un DataFrame de Pandas si se han extraído enlaces. Si no se han extraído enlaces, se imprime un mensaje indicando que no se realiza scraping.

***Botón 4, suele dar fallos, aunque sea visible, se decide, por tanto, aplicar el código de navegación.***

**Botón 4 a la 15: Este código se encarga de navegar a una página deseada y extraer datos de anuncios en esa página.**

A continuación, explicaré en detalle cada parte del código:

1. driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);").Realiza un scroll en la página.Este script de JavaScript se ejecuta para desplazar la página hacia abajo y asegurarse de que el enlace a la página siguiente sea visible
2. pagina\_deseada y pagina\_actual: Estas variables se utilizan para definir la página deseada a la que se quiere llegar (en este caso, la página 4) y realizar un seguimiento de la página actual durante la navegación.
3. while pagina\_actual < pagina\_deseada:: Esto inicia un bucle while que continuará hasta que la pagina\_actual sea igual a la pagina\_deseada. El propósito de este bucle es navegar a través de las páginas haciendo clic en el botón "Siguiente" hasta llegar a la página deseada.
4. try:: Se inicia un bloque try para manejar posibles excepciones durante la navegación.
5. boton\_siguiente = WebDriverWait(driver, 10).until(...): Este código utiliza WebDriverWait para esperar hasta que el botón "Siguiente" esté presente en el DOM y sea clickeable. Utiliza una expresión XPath para encontrar el botón que contiene el atributo 'aria-label' con el valor 'Siguiente'.
6. boton\_siguiente = driver.find\_element(By.XPATH, "//a[contains(@class, 'atm\_1y33qqm\_1ggndnn') and contains(text(), 'Siguiente')]"): En esta línea de código, se busca y se selecciona el botón "Siguiente" en la página actual utilizando una expresión XPath. El XPath se utiliza para localizar elementos en el DOM del navegador. En este caso, se busca un elemento <a> (hipervínculo) que contenga ciertas clases y texto específico. Si se encuentra el botón "Siguiente", se almacena en la variable boton\_siguiente.
7. time.sleep(5): Se espera 5 segundos después de hacer clic en el botón "Siguiente" para permitir que la página se cargue completamente antes de continuar.
8. boton\_siguiente.click(): Luego, se hace clic en el botón "Siguiente" para avanzar a la siguiente página de resultados
9. pagina\_actual += 1: Se incrementa la variable pagina\_actual en 1 para realizar un seguimiento de la página actual.
10. print(f"Avanzando a la página {pagina\_actual + 1}"): Se imprime un mensaje que indica a qué página se está avanzando.
11. print("URL actual:", driver.current\_url): Se imprime la URL actual después de cambiar de página.
12. except Exception as e:: En caso de que no se pueda hacer clic en el botón "Siguiente" (por ejemplo, si ya no hay más páginas), se captura una excepción y se imprime un mensaje. Luego, se rompe el bucle con break para salir del bucle while.
13. Después de salir del bucle, se verifica si pagina\_actual es igual a pagina\_deseada. Si son iguales, significa que se ha llegado a la página deseada y se procede a extraer datos de los anuncios en esa página.
14. Se crea una lista vacía llamada anuncios\_pisos\_pag4 para almacenar la información de los anuncios en la página 4.
15. Se utilizan expresiones WebDriverWait para esperar a que los elementos de los anuncios estén presentes en el DOM antes de extraer datos. Estos elementos incluyen títulos, subtítulos, valoraciones, precios por noche, precios totales, entre otros.
16. Se recorren todos los elementos de anuncios en la página deseada utilizando un bucle for, y se extrae información de cada uno de ellos, incluyendo enlaces y atributos relevantes.
17. Se maneja la posibilidad de que no se puedan encontrar algunos elementos y se asigna 'N/A' en caso de que falten datos.
18. Se intenta realizar scraping de características específicas de cada anuncio, similar a lo explicado en bloques de código anteriores.
19. Si se obtienen enlaces y se extrae información de los anuncios, se crea un DataFrame de Pandas (df) con los datos de los anuncios.
20. Si no se obtienen enlaces en la página deseada, se imprime un mensaje indicando que no se realiza scraping.

**En general y en cada botón**

El fragmento de código extracción de elementos página principal:

try: # Realizar clic en el botón Siguiente

WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, 'c4mnd7m'))) # Esperar hasta que se carguen todos los elementos de anuncios en el DOM (10 segundos)

elementos\_anunciospag4 = driver.find\_elements(By.CLASS\_NAME, 'c4mnd7m') # Encontrar todos los elementos de anuncios en la página deseada

# Esperas para elementos adicionales en el DOM

precios\_noche = WebDriverWait(driver, 30).until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, '\_1jo4hgw'))) # Esperar hasta que se carguen todos los elementos de anuncios en el DOM (30 segundos)

subtitulos\_anuncio = WebDriverWait(driver, 30).until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, 't6mzqp7')))# Esperar hasta que se carguen todos los elementos de anuncios en el DOM (30 segundos)

valoraciones\_anuncio = WebDriverWait(driver, 30).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, ".r1dxllyb.atm\_7l\_18pqv07.atm\_cp\_1ts48j8.dir.dir-ltr")))# Esperar hasta que se carguen todos los elementos de anuncios en el DOM (30 segundos)

titulos\_anuncio = WebDriverWait(driver, 30).until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, 't1jojoys')))# Esperar hasta que se carguen todos los elementos de anuncios en el DOM (30 segundos)

precios\_total = WebDriverWait(driver, 30).until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, '\_tt122m')))# Esperar hasta que se carguen todos los elementos de anuncios en el DOM (30 segundos)

for index, anunciopag4 in enumerate(elementos\_anunciospag4): # Recorrer todos los elementos de anuncios en la página deseada. Index es el índice de cada elemento . Enumerate es el índice de cada elemento

enlace\_anuncio = anunciopag4.find\_element(By.TAG\_NAME, 'a') # Encontrar el enlace del anuncio

href\_anuncio = enlace\_anuncio.get\_attribute('href') # Extraer la URL del enlace

target\_anuncio = enlace\_anuncio.get\_attribute('target') # Extraer el atributo target

# Extraer información usando el índice correspondiente

titulo = titulos\_anuncio[index].text if index < len(titulos\_anuncio) else 'N/A' # Extraer el texto de la etiqueta si existe

subtitulo = subtitulos\_anuncio[index].text if index < len(subtitulos\_anuncio) else 'N/A' # Extraer el texto de la etiqueta si existe

valoracion = valoraciones\_anuncio[index].text if index < len(valoraciones\_anuncio) else 'N/A' # Extraer el texto de la etiqueta si existe

precio\_noche = precios\_noche[index].text if index < len(precios\_noche) else 'N/A' # Extraer el texto de la etiqueta si existe

precio\_total = precios\_total[index].text if index < len(precios\_total) else 'N/A' # Extraer el texto de la etiqueta si existe

anuncios\_pisos\_pag4.append({ #append es para agregar un elemento a la lista

'texto': anunciopag4.text,

'titulo': titulo,

'subtitulo': subtitulo,

'valoracion': valoracion,

'precio\_noche': precio\_noche,

'precio\_total': precio\_total,

'href': href\_anuncio,

'target': target\_anuncio

})

**Explicación General**:

1. **Inicialización de la Lista**: Antes de comenzar a extraer información de los anuncios en la página actual, se crea una lista vacía llamada anuncios\_pisos\_pag\_actual. Esta lista se utilizará para almacenar la información de cada anuncio en la página actual.
2. **Espera de Elementos en el DOM**: El código utiliza la función WebDriverWait de Selenium para esperar hasta que ciertos elementos estén presentes en el DOM del navegador. Estos elementos incluyen los anuncios principales y otros elementos relevantes como precios, subtítulos, valoraciones, títulos y precios totales. La espera se establece en un tiempo límite (por ejemplo, 10 o 30 segundos) para asegurarse de que la página haya cargado completamente y todos los elementos necesarios estén disponibles para su procesamiento.

elementos\_anunciospag\_actual = WebDriverWait(driver, 10).until(

EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, 'c4mnd7m'))

)

Además de los elementos principales, se espera a que otros elementos relevantes, como precios por noche, subtítulos, valoraciones, títulos y precio total, estén presentes en el DOM de la página:

precios\_noche = WebDriverWait(driver, 30).until(

EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, '\_1jo4hgw'))

)

subtitulos\_anuncio = WebDriverWait(driver, 30).until(

EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, 't6mzqp7'))

)

valoraciones\_anuncio = WebDriverWait(driver, 30).until(

EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, ".r1dxllyb.atm\_7l\_18pqv07.atm\_cp\_1ts48j8.dir.dir-ltr"))

)

titulos\_anuncio = WebDriverWait(driver, 30).until(

EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, 't1jojoys'))

)

precio\_total = WebDriverWait(driver, 30).until(

EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, '\_tt122m'))

)

Esto asegura que la página haya cargado completamente y todos los elementos necesarios estén disponibles para su procesamiento.

1. Luego, se inicia un bucle for para recorrer todos los elementos de anuncio en la página actual. Esto se hace con la ayuda de enumerate, que proporciona un índice para cada elemento en la lista elementos\_anunciospag\_actual. El índice se usa para acceder a los elementos correspondientes en las listas de precios, subtítulos, valoraciones, títulos y precio total. Aquí se procesa cada anuncio en detalle:

for index, anunciopag\_actual in enumerate(elementos\_anunciospag\_actual):

try:

# Extraer información de cada elemento del anuncio

```

En este bucle for, se recorren todos los elementos de anuncio en la página actual (no necesariamente la página 2, ya que el código generaliza para cualquier página). A continuación, explicaré cada parte del bucle en detalle:

1. for index, anunciopag\_actual in enumerate(elementos\_anunciospag\_actual):: Este bucle for utiliza la función enumerate para recorrer la lista de elementos\_anunciospag\_actual. La función enumerate asigna un índice (index) a cada elemento de la lista y lo almacena en la variable anunciopag\_actual. Esto permite recorrer la lista y, al mismo tiempo, realizar un seguimiento del índice del elemento actual.
2. try:: Se inicia un bloque try, que se utiliza para manejar cualquier excepción que pueda ocurrir durante el procesamiento de un anuncio.
3. # Extraer información de cada elemento del anuncio: Aquí se comentó una línea de código que vendría a continuación. En esta parte, se accede a cada elemento del anuncio (como el título, subtítulo, valoración, etc.) dentro de anunciopag\_actual y se extrae su contenido.
4. El código dentro del bloque try se encarga de extraer información específica de cada elemento del anuncio en la página actual, como el título, el subtítulo, la valoración, el precio por noche, el precio total, el enlace y el target del anuncio.
5. Cada una de estas piezas de información se extrae utilizando métodos y propiedades de los elementos HTML dentro de anunciopag\_actual. Por ejemplo, titulo\_anuncio se extrae del elemento titulos\_anuncio[index], y lo mismo se aplica a las otras variables.
6. Los valores extraídos se almacenan en las variables correspondientes (titulo\_anuncio, subtitulo\_anuncio, valoración\_anuncio, precio\_noche, precio\_total\_texto, href\_anuncio, target\_anuncio).
7. A continuación, se imprimen estos valores en la consola para que el usuario pueda verlos durante la ejecución del programa. Esto puede ser útil para el proceso de depuración y seguimiento del progreso del scraping.
8. Finalmente, todos estos valores se almacenan en un diccionario y se agregan a la lista anuncios\_pisos\_pag\_actual como un elemento del diccionario. El diccionario almacena todos los detalles del anuncio, y la lista anuncios\_pisos\_pag\_actual se usa para recopilar estos diccionarios para cada anuncio en la página actual.
9. titulo\_anuncio = titulos\_anuncio[index].text if index < len(titulos\_anuncio) else 'N/A'
   * + Sirve para extraer el título de un anuncio de una página web y almacenarlo en la variable titulo\_anuncio. Vamos a desglosarla paso a paso:
     + titulos\_anuncio: Esto es una lista que contiene todos los elementos HTML que tienen la clase "t1jojoys" en la página web. En el contexto de tu código, estos elementos son los títulos de los anuncios que estás buscando.
     + index: Es una variable que está siendo utilizada en un bucle for para recorrer los elementos de titulos\_anuncio. En cada iteración, index tomará el valor del índice actual del bucle.
     + titulos\_anuncio[index]: Esto accede al elemento de la lista titulos\_anuncio en la posición dada por index. En otras palabras, obtiene el título del anuncio en la posición actual del bucle.
     + .text: Esto es una propiedad de un elemento HTML en Selenium que te permite obtener el texto contenido dentro de ese elemento. En este caso, se está extrayendo el texto del título del anuncio.
     + if index < len(titulos\_anuncio) else 'N/A': Esta es una expresión condicional que se asegura de que la extracción del título sea segura. Verifica si el valor de index es menor que la longitud de la lista titulos\_anuncio. Si es cierto, significa que hay un título en la posición actual y se asigna ese título a titulo\_anuncio. Si es falso, lo que significa que no se encontró un título en esa posición, se asigna 'N/A' (que significa "No disponible") a titulo\_anuncio.

En resumen, esta línea de código se utiliza para extraer el título de un anuncio de la página web de Airbnb. Si se encuentra un título válido en la posición actual, se almacena en la variable titulo\_anuncio. Si no se encuentra un título válido, se asigna 'N/A' como un valor por defecto para indicar que no se encontró ningún título en esa posición.

En resumen, este bucle for recorre cada elemento de anuncio en la página actual, extrae información relevante de esos elementos y almacena esos datos en una estructura de datos (anuncios\_pisos\_pag\_actual) para su posterior procesamiento o almacenamiento. La utilización de enumerate permite un acceso ordenado a los elementos mientras se realiza un seguimiento de su posición en la lista.

El fragmento de código extracción p de características adicionales de cada anuncio haciendo click en los enlaces:

# Intentar realizar scraping solo si se obtuvieron enlaces

if anuncios\_pisos\_pag4: # Verificación de que se hayan extraído anuncios

try:

for anuncio in anuncios\_pisos\_pag4: # Recorrer todos los anuncios

driver.get(anuncio['href']) # Ir a la página del anuncio extraído

WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "lgx66tx.atm\_gi\_idpfg4.atm\_l8\_idpfg4.dir.dir-ltr"))) # Esperar a que se carguen las características

caracteristicas = driver.find\_elements(By.CLASS\_NAME, "lgx66tx.atm\_gi\_idpfg4.atm\_l8\_idpfg4.dir.dir-ltr") # Encontrar todas las características

caracteristicas\_texto = [c.text for c in caracteristicas] # Extraer el texto de las características

print(f"Características para {anuncio['href']}: {caracteristicas\_texto}") # Imprimir las características

anuncio['caracteristicas'] = caracteristicas\_texto # Agregar las características al anuncio

except Exception as e: # En caso de error

print(f"No se pudo realizar scraping de características para {anuncio['href']}: {e}")

Ahora, desglosamos cómo funciona esta parte del código:

1. **Verificación de Enlaces**: Primero, se verifica si la lista anuncios\_pisos\_pag contiene enlaces válidos. Esto se hace con la condición if anuncios\_pisos\_pag:. Si la lista no está vacía (es decir, si se han extraído anuncios), el código procederá a realizar el scraping de características adicionales.
2. **Recorriendo Anuncios y Accediendo a las Páginas de Anuncios**: Se utiliza un bucle for para recorrer cada anuncio en la lista anuncios\_pisos\_pag. Dentro del bucle, se obtiene el enlace (href) de cada anuncio y se navega a la página de ese anuncio utilizando driver.get(anuncio['href']).
3. **Espera de las Características**: Después de cargar la página del anuncio, el código espera a que se carguen las características adicionales. Esto se hace mediante WebDriverWait y la función EC.presence\_of\_all\_elements\_located. El código espera a que los elementos con la clase lgx66tx.atm\_gi\_idpfg4.atm\_l8\_idpfg4.dir.dir-ltr estén presentes en el DOM antes de continuar.
4. **Extracción de Características**: Una vez que los elementos de las características adicionales están presentes, se utilizan driver.find\_elements para encontrar todos estos elementos. Luego, se extrae el texto de cada elemento y se almacena en la lista caracteristicas\_texto. Estos textos representan las características adicionales del anuncio.
5. **Almacenamiento de Características**: Se imprime en la consola el texto de las características y se agrega a la clave caracteristicas del diccionario del anuncio correspondiente en la lista anuncios\_pisos\_pag. Esto permite almacenar las características adicionales junto con la información básica del anuncio.
6. **Manejo de Errores**: Se incluye un bloque except para manejar cualquier excepción que pueda ocurrir durante el scraping de características. Si hay un error, se imprime un mensaje en la consola indicando que no se pudo realizar el scraping de características para ese anuncio específico.

En resumen, esta parte del código se encarga de extraer características adicionales de cada anuncio en la página actual, siempre que haya enlaces válidos en la lista de anuncios. Cada conjunto de características se almacena junto con la información básica del anuncio en la lista anuncios\_pisos\_pag.

## CREAR DF Y GUARDAR EN CSV. FORMATO TEXTO

# Lista principal para almacenar la información de todos los anuncios

todos\_los\_anuncios = []

todos\_los\_anuncios.extend(anuncios\_pisos\_pag1)

……………………

# Crear un DataFrame a partir de la lista de todos los anuncios

df\_anuncios = pd.DataFrame(todos\_los\_anuncios)

# Limpiar los saltos de línea y comas en las columnas de texto

df\_anuncios['titulo'] = df\_anuncios['titulo'].str.replace('\n', ' ').str.replace(',', '')

df\_anuncios['subtitulo'] = df\_anuncios['subtitulo'].str.replace('\n', ' ').str.replace(',', '')

# Repetir para cualquier otra columna que pueda tener este problema

# Ruta donde deseas guardar el archivo CSV (nombre único con marca de tiempo)

timestamp = datetime.now().strftime("%Y%m%d%H%M%S")

nombre\_archivo\_unico = f'anuncios\_airbnb\_girona\_{timestamp}.csv'

ruta\_guardado = f'/content/drive/MyDrive/Hosting/Hosting/Analisis Scrapping/csv/{nombre\_archivo\_unico}'  # Ajusta esta ruta según sea necesario

# Guardar el DataFrame en un archivo CSV único basado en la marca de tiempo

df\_anuncios.to\_csv(ruta\_guardado, sep='\t', encoding='utf-16', index=False)

print(f"Archivo CSV único con marca de tiempo guardado en: {ruta\_guardado}")

Este bloque de código tiene el propósito de consolidar información de varios conjuntos de datos, limpiar esa información, y luego almacenarla en un archivo CSV. Aquí hay una explicación de cada línea:

* Se crea una lista vacía llamada todos\_los\_anuncios, que almacenará información de múltiples páginas.
* La lista se extiende con los datos de las variables anuncios\_pisos\_pagX, que aparentemente contienen anuncios de diferentes páginas (del 1 al 15) de una fuente como Airbnb.
* Se crea un DataFrame de pandas llamado df\_anuncios a partir de la lista combinada de todos los anuncios.
* Se limpian los títulos y subtítulos de los anuncios en el DataFrame para eliminar saltos de línea y comas, lo que podría afectar el formato del CSV.
* Se genera un nombre de archivo único para el CSV utilizando la fecha y hora actual con un formato específico.
* Se establece la ruta de guardado del archivo CSV, la cual incluye el nombre del archivo único.
* Se guarda el DataFrame en un archivo CSV en la ruta especificada, utilizando una tabulación como separador y la codificación UTF-16.
* Finalmente, se imprime la ruta de guardado para notificar al usuario dónde se ha guardado el archivo CSV.

## 

### INFORMACIÓN ADICIONAL

Se ha incluido al análisis, los listings que se han descargado a través de la página <http://insideairbnb.com/get-the-data>, teniendo en cuenta que sólo podremos obtener los del año 2023, ya que es la única información pública que se puede obtener. Quedándose por tanto, fuera del análisis el año 2022.

## CONCLUSIONES

Esta sección resume los resultados del proyecto, los desafíos encontrados y las posibles mejoras para futuras tareas de web scraping.

Desafíos:

1. **Página Dinámica:** Airbnb utiliza una página web dinámica, lo que significa que su contenido cambia cada vez que haces clic en ella o interactúas con elementos como botones de carga o paginación.
2. **Límite de Extracción:** Airbnb limita la cantidad de anuncios que puedes extraer en una sola solicitud a aproximadamente 270. Esto hace que sea necesario ejecutar el código varias veces para recopilar una gran cantidad de datos. Además, la información que se puede extraer corresponde al año actual, 2024. Tal y como se ha comentado en punto 8 de este informe, se ha obtenido más información de los listings del último año en la página <http://insideairbnb.com/get-the-data>, quedando el año 2022 sin información en este análisis.
3. **Evolución del Sitio:** Los CSS pueden cambiar con el tiempo debido a actualizaciones del sitio, lo que puede romper el código de scraping existente.
4. **Restricciones de Request:** Airbnb puede bloquear o limitar el acceso a su sitio web cuando se realizan demasiadas solicitudes en un corto período de tiempo. Esto hace que sea necesario utilizar Selenium (driver) en lugar de la biblioteca requests para simular interacciones más humanas y evitar bloqueos.
5. **Manejo de WebDriver y ChromeDriver:**

* **Errores de Inicialización:** Problemas al inicializar el WebDriver de Selenium, específicamente con ChromeDriverManager, incluyendo errores como AttributeError y TypeError. Esto podría deberse a problemas con la versión específica del ChromeDriver o problemas al intentar instalarlo automáticamente.
* **Configuración del Navegador:** Dificultades para configurar las opciones del navegador Chrome, como el modo headless y otros parámetros de inicialización.

1. **Selección de Elementos y Esperas:**

* **Localización de Elementos**: Problemas al encontrar y seleccionar elementos en la página utilizando Selenium, debido a cambios potenciales en la estructura del DOM o en los selectores de clases utilizados.
* **Esperas Implícitas y Explícitas**: Errores al usar WebDriverWait para esperar a que los elementos estén presentes en el DOM, y problemas con elementos obsoletos o referencias estancadas.

1. **Navegación de las páginas ocultas:**
2. **Uso de Try except para evitar que el código se pare.**
3. **Se han creado varios códigos**

En resumen, el web scraping en Airbnb presenta desafíos debido a su naturaleza dinámica y a las limitaciones de extracción impuestas. Para abordar estos desafíos, es importante utilizar herramientas y estrategias adecuadas y mantenerse atento a las posibles actualizaciones del sitio.

# 5. PREGUNTAS CLIENTE

El objetivo principal de nuestro análisis de datos es proporcionar respuestas sólidas a las preguntas clave planteadas por nuestro cliente, Entendemos que WhiteHosting busca tomar decisiones informadas y estratégicas sobre la asignación de capital en propiedades inmobiliarias. Para lograr esto, hemos respondido cuidadosamente a una serie de preguntas específicas formuladas por WhiteHosting. Estas preguntas abordan aspectos fundamentales del mercado inmobiliario y buscan insights valiosos para orientar las decisiones de inversión.

En este contexto, nos enfocaremos en abordar cada una de las preguntas del cliente de manera exhaustiva y detallada. Utilizando técnicas de análisis de datos mencionadas anteriormente, exploraremos conjuntos de datos tanto internos como externos, incluidos los datos proporcionados por WhiteHosting y los datos extraídos de fuentes externas (Airbnb). A través de nuestro análisis, buscaremos identificar patrones, tendencias y oportunidades en el mercado inmobiliario que puedan ser de interés para WhiteHosting.

A continuación, procederemos a explorar cada una de las preguntas planteadas por WhiteHosting, abordando los aspectos clave del análisis de datos que nos permitirán ofrecer respuestas claras y fundamentadas.

## ¿Cuántos inmuebles únicos hay en el dataset vs AirBnb?

* Inmuebles únicos en el df original Whitehosting: 8796
* Inmuebles únicos en el df AirBnb 2023: 66181
* Inmuebles únicos en el df AirBnb 2024: 2837

TOTAL INMUEBLES ÚNICOS ENTRE LOS 4: 77.814

## ¿Cuál es la diferencia en número de inmuebles posteados entre las dos fuentes de datos?

La diferencia entre el número de inmuebles únicos posteados entre el dataset original y los extraídos de AirBnb es de 60.222

## ¿Cuáles son los parámetros que conforman el precio?¿Son iguales los parámetros en ambas fuentes de datos?

Para comprobar qué parámetros conforman el precio se ha llevado a cabo el cálculo de la correlación existente entre el precio y las demás variables, obteniendo como resultado lo siguiente:

* DATASET ORIGINAL: En el dataset original las variables con un coeficiente de correlación mayor con el precio son, en orden de mayor a menor:

‘acommodates’, ‘bedrooms’, ‘bathrooms’, ‘bed’.

* AIRBNB: En los dataset de AirBnb las variables con un coeficiente de correlación mayor con el precio son, en orden de mayor a menor:

‘acommodates’, ‘bedrooms’, ‘bathrooms’, ‘bed’.

Por lo tanto, en ambas fuentes de datos, aun dando cifras diferentes, debido, seguramente, a diferencias en los datos recopilados, los parámetros que conforman el precio son los mismos.

En el primer análisis de la correlación se observó que en las valoraciones de los huéspedes no tenían correlación con el precio.

## ¿Has podido ver algún inmueble que esté en ambas fuentes de datos?¿Tienen la misma valoración?

**Metodología de Análisis:**

1. **Identificación de Coincidencias**: Se implementaron varias estrategias para identificar inmuebles coincidentes entre los DataFrames de diferentes años (2021, 2023 y 2024), utilizando tanto 'apartment\_id' cómo 'license' como claves de búsqueda. Esto permitió determinar la presencia de inmuebles en más de un conjunto de datos. Finalmente, se decidió mantener la variable “apartament\_id” para seguir con el análisis ya que “license” sólo encontraba coincidencias con 2021 y 2023.
2. **Comparación de Valoraciones**: Para los inmuebles identificados como comunes entre los DataFrames, se procedió a comparar sus valoraciones para determinar si estas eran consistentes a través del tiempo. Esto implicó verificar si las valoraciones de los inmuebles en cada año eran iguales o difieren. Como para el df original la valoración era sobre 100 se procedió a normalizar el dato sobre 5 para que ambos dataframes fueran comparables.

**Conclusiones Clave:**

* **Presencia de Inmuebles Comunes**: Se confirmó la existencia de inmuebles que aparecen en más de uno de los DataFrames analizados. Esto indica que algunos inmuebles han sido listados en Airbnb en múltiples ocasiones a lo largo de los años considerados.
* **Desafíos y Limitaciones**: La investigación enfrentó desafíos inherentes a la calidad y consistencia de los datos, como la ausencia de la columna 'license' en algunos DataFrames, en 2024, y las variaciones en la cantidad de reseñas por año. Estos aspectos subrayan la complejidad de realizar análisis longitudinales en conjuntos de datos dinámicos como los de Airbnb.
* **Variabilidad en las Valoraciones**: El análisis detallado de los inmuebles en Airbnb revela que, si bien hay consistencia en las valoraciones para algunos apartamentos, existe una variabilidad significativa para otros. Esta dinámica resalta la naturaleza volátil del mercado de alquileres vacacionales, donde las percepciones y experiencias de los huéspedes son susceptibles a una amplia gama de factores. Estos pueden incluir desde mejoras en los inmuebles y cambios en la gestión, hasta variaciones en las tendencias del mercado y las expectativas de los clientes.

**Conclusión General**:

La variabilidad en las valoraciones a lo largo del tiempo enfatiza la necesidad de una gestión dinámica y proactiva de los inmuebles listados en Airbnb. Los propietarios y administradores deben enfocarse no solo en mantener sus propiedades en un estado excelente, sino también en adaptarse a las tendencias cambiantes del mercado y a las expectativas evolutivas de los huéspedes. Para los analistas de mercado, estos patrones de valoración subrayan los desafíos de predecir el rendimiento de los inmuebles basándose solamente en datos históricos. Es crucial incorporar un análisis multifacético que considere factores económicos, sociales y competitivos para obtener una comprensión integral del posicionamiento y potencial de un inmueble dentro del ecosistema de alojamiento vacacional.

## 5. ¿En qué fuente de datos hay más volatilidad en cuanto al precio?¿Hay algún tipo de estacionalidad?

El análisis se basa en la medición de la volatilidad mediante la desviación estándar de los precios en diferentes ciudades para cada año. Este indicador proporciona una medida de la variabilidad de precios en relación con su media. La volatilidad se midió calculando la desviación estándar de los precios de los alojamientos en diferentes ciudades para cada año. Este índice proporciona una medida de la dispersión de precios en relación con su media.

* Volatilidad por Ciudad en 2021: Se observó una amplia variabilidad en la volatilidad entre las ciudades, con Menorca, Mallorca y Sevilla mostrando la mayor volatilidad, mientras que Valencia, Málaga y Madrid exhibieron una volatilidad más baja.
* Aumento General en 2023: La volatilidad experimentó un aumento notable en todas las ciudades en 2023 en comparación con 2021, lo que puede atribuirse a factores como la recuperación post-COVID-19 o impactos económicos globales.
* Estabilización en 2024: Aunque la volatilidad se mantuvo alta en ciudades como Barcelona y Madrid en 2024, se observó una tendencia hacia la estabilización en general.

La interpretación de la Volatilidad y Estacionalidad que obtuvimos fue:

* Factores Influenciadores: La estacionalidad turística, eventos locales y nacionales, y las fluctuaciones económicas pueden influir en la volatilidad de los precios.
* Consideración de Variables Complementarias: Es esencial examinar la volatilidad junto con otras variables como el precio promedio y la tasa de ocupación para obtener una comprensión más completa del mercado.
* Uso Estratégico del Análisis: La comprensión de la volatilidad puede beneficiar a los anfitriones de Airbnb para ajustar estratégicamente los precios y a los viajeros para planificar sus viajes de manera más informada.

Con las siguientes limitaciones y consideraciones adicionales

* Cobertura de Datos: El análisis está limitado a los datos disponibles para los años específicos y no incluye información para el año 2022, lo que puede afectar la precisión de la interpretación a largo plazo.
* Exclusión del DF Scraping: La ausencia de datos del DF Scrapping debido a la falta de información sobre la variable "Last\_review" limita el alcance del análisis y podría proporcionar insights adicionales sobre la volatilidad del mercado.

La conclusión es que la volatilidad de precios en el mercado de alojamientos vacacionales ha mostrado una evolución significativa en el período analizado. Se recomienda continuar monitoreando la volatilidad para adaptar estrategias de precios y anticipar las tendencias del mercado. Además, se sugiere ampliar la cobertura de datos y considerar la inclusión de datos adicionales para futuros análisis.

## 6. Crea un dataset único

Antes de comenzar, vamos a ofrecer una visión general sobre el análisis exploratorio de datos (EDA) realizado en los conjuntos de datos de inside\_airbnb con información del año 2023, y de scrapping\_airbnb con datos del 2024:

### **EDA para inside\_airbnb 2023**

Para construir este conjunto de datos, se descargaron los archivos listing.csv correspondientes a las mismas ciudades incluidas en el conjunto de datos original proporcionado por el cliente. Posteriormente, se llevó a cabo la normalización de campos y variables pertinentes, así como el análisis estadístico de las variables que determinan los parámetros de precio. Es notable que este conjunto de datos abarca un alto número de alojamientos, ya que incluye todos los listados en Airbnb durante el 2023, aunque es importante mencionar que solo se refiere a alojamientos vacacionales. Se identificaron propiedades consideradas de lujo, cuyos precios elevados se deben a que son alquileres estacionales, principalmente durante el verano. Para el tratamiento de valores atípicos, se optó por diferentes análisis. Primero, se calculó el rango intercuartílico (IQR) para 'price' y se integraron análisis con todos los conjuntos de datos. Se observó que, aunque el 2023 tuvo la mayor cantidad de datos, la volatilidad de los precios fue menor comparada con los años 2021 y 2024. Esto se debe a dos factores: la eliminación de numerosos alojamientos valiosos en el análisis del conjunto de datos inside\_airbnb 2022 y a que en los conjuntos de datos originales de 2021 y scrapping\_airbnb 2024 no se eliminaron los valores atípicos. Por lo tanto, se decidió realizar análisis adicionales en el conjunto de datos inside\_airbnb y finalmente conservar un conjunto de datos excluyendo las propiedades de lujo para permitir comparaciones adecuadas con los otros dos conjuntos de datos.

Motivos para no eliminar los valores atípicos del precio

Se introdujo una nueva columna llamada "precio\_noche" para intentar estandarizar los precios dentro de un mismo rango de noches. Sin embargo, los resultados no fueron considerados válidos ya que el precio mínimo era inferior a 0, y el promedio de precios disminuyó significativamente.

Los precios que parecían ser valores atípicos podrían representar los costos reales de habitaciones de hotel durante la temporada alta. Tras verificarlo, se confirmó que coinciden con la temporada alta. Eliminar estos datos podría sesgar el análisis y conducir a conclusiones erróneas.

Por lo tanto, se decidió mantener los valores atípicos, filtrando aquellos con un precio superior a 5000 euros. Así, se excluyen las propiedades consideradas de lujo.

### **EDA para scrapping\_airbnb 2024**

Para elaborar el conjunto de datos scrapping\_airbnb 2024, se realizaron al menos dos descargas de listados por ciudad, incluyendo todas las ciudades presentes en el conjunto de datos original. Se efectuó la normalización de campos y variables, y se eliminaron los valores nulos. También se analizaron las estadísticas de las variables relacionadas con los precios, notando que los inmuebles con precios altos generalmente corresponden a villas, las cuales no se consideran valores atípicos.

### **Creación del dataset único**

La tarea de realizar la unión de los diferentes conjuntos de datos de dónde procedencia los datos para el proyecto Hosting (listings\_airbnb\_original, listings\_inside\_ airbnb\_2023, y listings\_scrapping\_airbnb\_2024) en un DataFrame unificado presenta tanto oportunidades como desafíos significativos. A continuación, se presentan las conclusiones y la propuesta basada en el proceso descrito:

### **Conclusiones sobre el Proceso de Unión**

1. Conversión de Tipos de Datos: La decisión de convertir todas las columnas de los Data Frames a tipo string facilita la unión y manejo de datos, especialmente cuando se trata de valores no numéricos o datos mixtos. Sin embargo, esto podría limitar el análisis posterior que requieran operaciones numéricas o temporales específicas, por lo que se recomienda convertir de nuevo ciertas columnas a sus tipos originales cuando sea necesario.
2. Unión de DataFrames: La estrategia de unir primero el Data Frame original con listings\_airbnb\_2023 utilizando un conjunto específico de columnas comunes, seguido de la inclusión de listings\_airbnb\_2024, intenta crear un conjunto de datos completo que abarque múltiples años. No obstante, la utilización de pd.concat() seguida de pd.merge() para agregar listings\_airbnb\_2024 podría generar redundancias o conflictos en los datos debido a diferencias en la estructura y contenido de los DataFrames.
3. Generación de filas NaN: La unión de Data Frames con diferentes conjuntos de columnas puede resultar en un número significativo de filas con valores NaN, especialmente si el Data Frame original contiene más columnas únicas que los otros. Esto no solo aumenta la complejidad del conjunto de datos sino que también puede afectar negativamente el rendimiento y la usabilidad en herramientas de análisis como Power BI.

### **Propuesta para Manejar los Desafíos**

Para abordar los desafíos identificados y optimizar el conjunto de datos para análisis posteriores y visualizaciones en Power BI, se propone la siguiente estrategia:

Creación de una Tabla de Hechos: Identificar y utilizar un conjunto de columnas comunes a los tres DataFrames para formar una "Tabla de Hechos" central. Esta tabla debería enfocarse en datos clave como 'apartment\_id', 'price', 'city', 'year\_df', 'bedrooms', 'beds', y 'bathrooms', que son esenciales para análisis comparativos y temporales.

Tablas de Dimensiones Adicionales: Para las columnas que no son comunes entre todos los Data Frames pero ofrecen información valiosa (por ejemplo, características únicas del inmueble, valoraciones, etc.), se deberían crear tablas de dimensiones adicionales. Estas tablas pueden vincularse a la Tabla de Hechos mediante claves únicas (como 'apartment\_id') para enriquecer los análisis sin sobrecargar la Tabla de Hechos con datos faltantes.

Minimización de Datos Faltantes: Al centrarse en las columnas comunes para la Tabla de Hechos y delegar datos adicionales a tablas de dimensiones, se minimiza la cantidad de filas con valores NaN. Esto simplifica la limpieza de datos y mejora la integridad del conjunto de datos.

Optimización para Power BI: Esta estructura de datos, con una Tabla de Hechos central y tablas de dimensiones relacionadas, está bien adaptada para herramientas de visualización como Power BI. Permite un manejo eficiente de los datos, reduce el rendimiento afectado por grandes volúmenes de filas NaN, y facilita la creación de dashboards dinámicos y análisis multidimensionales.

### **Conclusión General**

El proceso de integrar múltiples conjuntos de datos de Airbnb en un marco unificado para análisis y visualización presenta desafíos que requieren consideraciones cuidadosas sobre la estructura de datos y la gestión de valores faltantes. Adoptar un enfoque basado en la creación de una Tabla de Hechos y tablas de dimensiones separadas ofrece una solución pragmática que equilibra la necesidad de integridad de datos con la usabilidad y rendimiento en herramientas analíticas avanzadas como Power BI.

## 7. ¿Hay alguna variable que podríamos añadir para mejorar el análisis?

### VARIABLE OCUPACIÓN

Para la variable "Ocupación", primero se calcula el número de días no disponibles para reserva en un año, que se obtiene restando el número de días disponibles en el año (365 días) del valor de la columna "availability\_365". Esto nos da el número de días reservados por año.

A continuación, se agrupan los alojamientos por ciudad y se calcula la media de días reservados para cada ciudad. Esto nos proporciona el promedio de días reservados en un año para cada ciudad.

Por último, se calcula el porcentaje de ocupación para cada ciudad dividiendo la media de días reservados entre 365 (el número total de días en un año) y multiplicando el resultado por 100. Esto nos da el porcentaje de ocupación promedio para cada ciudad.

Los resultados muestran el porcentaje promedio de ocupación para cada ciudad en el año, lo que proporciona una idea de la demanda y popularidad de los alojamientos en esas ciudades. Un mayor porcentaje de ocupación indica una mayor demanda y una mayor tasa de reserva de alojamientos en esa ciudad, lo que puede ser útil para comprender la dinámica del mercado y tomar decisiones comerciales relacionadas con la inversión en bienes raíces para alquiler vacacional.

| CIUDAD | OCUPACIÓN |
| --- | --- |
| BARCELONA | 51,58 |
| GIRONA | 46,36 |
| MADRID | 55,31 |
| MÁLAGA | 44,93 |
| MALLORCA | 41,62 |
| MENORCA | 45,69 |
| SEVILLA | 46,35 |
| VALENCIA | 54,05 |

VARIABLE RENTABILIDAD

La variable que llamamos "Rentabilidad" en realidad representa los ingresos generados por los alojamientos en cada ciudad, ya que no disponemos del dato sobre el costo de adquisición de los mismos. Por lo tanto, es importante aclarar que estamos calculando los ingresos obtenidos y no la rentabilidad en términos financieros clásicos.

Para calcular esta variable, multiplicamos el número de días reservados por el precio diario del alojamiento, lo que nos da los ingresos generados por reserva. Luego, agrupamos los alojamientos por ciudad y calculamos el ingreso promedio obtenido en cada ciudad.

El resultado es una tabla que muestra el ingreso promedio generado por los alojamientos en cada ciudad. Estos números son útiles para evaluar la rentabilidad relativa de los alojamientos en diferentes ciudades en términos de ingresos generados, lo que puede ser valioso para la toma de decisiones en el ámbito de la inversión en propiedades para alquiler vacacional.

Los datos muestran que ciudades como Menorca, Mallorca y Girona tienen los ingresos promedio más altos por alojamiento, lo que sugiere una mayor rentabilidad en términos de ingresos obtenidos en comparación con otras ciudades. Esto podría influir en la estrategia de inversión de una empresa como WhiteHosting, que busca maximizar sus ingresos a través de la adquisición de propiedades para alquiler vacacional.

| CIUDAD | RENTABILIDAD |
| --- | --- |
| BARCELONA | 27848 |
| GIRONA | 24200 |
| MADRID | 19741 |
| MÁLAGA | 15079 |
| MALLORCA | 14891 |
| MENORCA | 13755 |
| SEVILLA | 11701 |
| VALENCIA | 11618 |

VARIABLE ESTACIONALIDAD

La variable de estacionalidad se analizó utilizando los datos de la última reseña ("last\_review\_date") de los alojamientos. Primero, convertimos esta columna a formato de fecha y luego la dividimos en mes y año para comprender mejor la distribución de las revisiones en función del tiempo.

El histograma muestra la frecuencia de las reseñas realizadas en cada mes, lo que nos da una idea de la estacionalidad de las reservas de alojamiento. Observamos picos y valles en diferentes meses, lo que indica períodos de alta y baja actividad de reservas. Los picos más altos se dan en los meses de agosto, septiembre y octubre. Y los valle en abril, mayo y diciembre.

Además, se presenta un histograma por ciudad que muestra la distribución de las revisiones mensuales para cada ciudad, lo que nos permite comparar la estacionalidad entre diferentes ubicaciones. De mes con mayor pico, agosto, prácticamente una tercera parte pertenece a reseñas de alojamientos de la ciudad de Girona.

Este análisis de estacionalidad es útil para comprender los patrones de demanda a lo largo del año, lo que puede ser valioso para la gestión de precios, la planificación de estrategias de marketing y la toma de decisiones relacionadas con la inversión en propiedades para alquiler vacacional.

## 8. ¿Puedes hacer un ranking de los inmuebles más caros?

Para responder a la pregunta de cuáles son los inmuebles más caros, se realizó un análisis en dos partes: primero, se creó un ranking de los alojamientos más caros en general, y segundo, se hizo un ranking por ciudad para identificar el alojamiento más caro en cada una.

Ranking General de Inmuebles Más Caros

Se ordenó el conjunto de datos de forma descendente por precio sin distinguir por ciudad. Esto nos permitió identificar los 10 alojamientos con el precio más alto en el conjunto de datos. Curiosamente, vemos que el precio máximo en esta muestra es de 500 euros, y estos alojamientos están distribuidos en diferentes ciudades y barrios, mostrando que hay una diversidad en la ubicación de los alojamientos más caros. La mayoría son del tipo "Entire home/apt", indicando que los alojamientos completos tienden a tener precios más altos.

Ranking de Inmuebles Más Caros por Ciudad

Después, se ordenó el conjunto de datos primero por ciudad y luego por precio de forma descendente, agrupando por ciudad para seleccionar el alojamiento más caro de cada una. Esto nos permitió ver cuál es el alojamiento más caro en cada ciudad, todos con precios cercanos o igual a 500 euros, excepto en Menorca, donde el precio más alto es ligeramente inferior. De nuevo, la mayoría son del tipo "Entire home/apt", reafirmando que este tipo de alojamiento suele ser el más costoso.

Análisis Gráfico de la Distribución de Precios

Por último, se creó un gráfico de caja (boxplot) para visualizar la distribución de precios de los alojamientos por ciudad. Este gráfico es útil para identificar no solo los alojamientos con precios máximos (como se hizo en los análisis previos), sino también para observar la mediana, los cuartiles y la presencia de valores atípicos en cada ciudad. La distribución de precios nos ayuda a entender en qué ciudades se concentra un mayor número de alojamientos caros y cómo varía el rango de precios entre diferentes ubicaciones.

Este análisis ofrece una visión clara de los precios de alojamiento en diferentes ciudades, identificando dónde se encuentran los inmuebles más caros y cómo se distribuyen los precios entre las distintas ciudades, lo que puede ser de gran utilidad tanto para viajeros con presupuestos variados como para anfitriones que buscan posicionar sus propiedades en el mercado.

## 9. ¿Cuáles son los inmuebles más rentables?

De nuevo, tenemos que tener en cuenta que el análisis de rentabilidad, se tuvo que realizar teniendo en cuenta sólo los ingresos por apartamento, pues no disponíamos de más información.

Para identificar los inmuebles más rentables, se llevó a cabo un análisis considerando tanto la ocupación como la rentabilidad de los alojamientos. Primero, se creó un ranking de los alojamientos más ocupados por ciudad y tipo de habitación, teniendo en cuenta la cantidad de días reservados en promedio y el precio. Este ranking muestra los alojamientos que han tenido una alta ocupación, lo que puede indicar su popularidad y demanda entre los usuarios.

Luego, se calculó la rentabilidad de los alojamientos multiplicando los días reservados por el precio. Posteriormente, se obtuvo un ranking de los alojamientos con la rentabilidad más alta, lo que indica aquellos inmuebles que han generado mayores ingresos.

Además, se crearon gráficos para visualizar mejor esta información. Por ejemplo, se mostró un gráfico de dispersión que representa la rentabilidad frente a la ubicación de los alojamientos más rentables, permitiendo ver claramente qué ciudades y barrios concentran los inmuebles más lucrativos. Asimismo, se elaboró un gráfico de barras que presenta la rentabilidad promedio de los alojamientos por ciudad, brindando una visión general de la rentabilidad en cada ubicación. Estos análisis y gráficos ofrecen una comprensión más completa de los inmuebles más rentables, tanto a nivel general como por ciudad y tipo de alojamiento.

# 6. MODELO ESTRELLA

A continuación, se presenta el modelo de datos.

## TABLA DE HECHOS

La tabla de hechos contiene las métricas numéricas que se desean analizar, generalmente medidas cuantitativas que se agrupan y resumen a lo largo de las dimensiones.

| apartment\_id |
| --- |
| name |
| room\_type |
| bathrooms |
| bedrooms |
| beds |
| price |
| rating |
| city |
| month |
| year\_review |
| year\_df |
| bathrooms\_cat |
| bedrooms\_cat |
| beds\_cat |
| price\_cat |
| id\_bedroom\_cat |
| id\_beds\_cat |
| id\_price\_cat |
| id\_city |
| region |
| id\_roomtype |
| rating\_cat |
| pais |
| date |
| city\_date |
| dimcity\_fech.id\_city\_fech |
| Month\_review |
| year\_review |
| estacionalidad |
| dimregion.ID.region |
| rentabilidad |
| inmuebles |

## 

## TABLAS DE DIMENSIONES

Las tablas de dimensiones contienen atributos descriptivos que se utilizan para filtrar, agrupar o agregar datos en la tabla de hechos.

1. CALENDARIO

Proporciona información sobre la fecha y el tiempo.

Incluye atributos como Año, Mes, Trimestre, Semana año, Día semana, Mes y año.

1. DIM\_AEROPUERTOS

Contiene detalles sobre los aeropuertos, incluidos datos como aeropuerto, año, total de pasajeros, variación con respecto al año anterior, etc.

1. DIM\_CITYMARKET

Ofrece información sobre el mercado de ciudades, incluyendo estimaciones de lugares, personal empleado, estancia promedio, viajeros nacionales e internacionales, etc.

1. DIM\_FEATURES\_APARTAMENTS

Describe las características de los apartamentos, como ID de apartamento, disponibilidad, fechas de revisión, etc.

1. DIMBARRIOS\_RENTABLES

Proporciona detalles sobre los barrios rentables, como el nombre del barrio, tipo de habitación, precio por metro cuadrado, etc.

1. DIMBATHROOMS\_CAT, DIMBATHROOMS\_CAR, DIMBEDROOMS\_CAT, DIMBEDS\_CAT

Categorizan los baños, dormitorios y camas según diferentes criterios.

1. DIMCITY, DIMCITY\_FECHA

Contienen información sobre la ciudad y la fecha respectivamente.

1. FIMGASTOASOCIADOSCOMPRA

Ofrece detalles sobre los gastos asociados a la compra de inmuebles, como impuestos, tasaciones, etc.

1. DIMPRECIOSm2INMUEBLES

Describe los precios por metro cuadrado de los inmuebles, incluyendo detalles sobre el precio, la superficie, el número de miembros del hogar, etc.

1. DIMPRICE\_CAT

Categoriza los precios en diferentes rangos o categorías para facilitar el análisis y la comparación de precios.

1. DIMRATING\_CAT

Categoriza las calificaciones en diferentes niveles o categorías, lo que permite evaluar la calidad de los alojamientos de manera más precisa.

1. DIMREGION

Proporciona información detallada sobre las regiones geográficas donde se encuentran las ciudades, permitiendo análisis basados en ubicaciones específicas.

1. DIMROOM\_TYPE

Describe los diferentes tipos de habitaciones disponibles en los alojamientos, como "Entire home/apt", "Private room", "Hotel room", etc.

1. DIMTOURISM\_INSIGHTS\_SPENDING

Contiene datos relacionados con los gastos turísticos e insights sobre el comportamiento de los turistas, lo que puede ser útil para entender mejor las tendencias de viaje y el impacto en el sector de alojamientos.

1. DURACION\_ESTANCIA

Categoriza la duración de la estancia en diferentes intervalos de tiempo, lo que puede ser relevante para comprender los patrones de reserva y la preferencia de los huéspedes.

1. INMUEBLESDUPLICADOS

Identifica y gestiona los inmuebles duplicados dentro de ambos datasets, lo que contribuye a una gestión eficiente de los datos y evita redundancias.

1. INMUEBLESMISMAVALORACION

Agrupa los inmuebles que tienen la misma valoración en ambos datasets, lo que nos sirve para contestar a alguna de las preguntas formuladas por el cliente.

# 7. CONCLUSIONES

Para finalizar el trabajo de análisis de datos, hemos llegado a una serie de conclusiones y recomendaciones clave que pueden guiar las decisiones estratégicas en el ámbito de las inversiones inmobiliarias del mercado de alojamientos vacacionales.

Durante el análisis de precios por metro cuadrado, se observa una notable disparidad en los precios de los pisos con más de en las diferentes ciudades, a excepción de Valencia. En Valencia, los precios de los inmuebles, tanto de más de como de menos, muestran una diferencia menos significativa. Sin embargo, en ciudades como Madrid o Barcelona, la diferencia entre los precios de los pisos con más de es considerablemente mayor en comparación con los precios de los pisos más pequeños. Esto sugiere que puede ser más conveniente invertir en estudios pequeños de pocos metros cuadrados en ciudades como estas, mientras que por un precio similar se puede invertir en alojamientos de más metros en la ciudad de Valencia.

Es fundamental tener en cuenta que la rentabilidad calculada se basa en los ingresos generados por el precio de alquiler, lo que quiere decir que es una representación teórica y no real de los posibles beneficios obtenidos. Ya que no estamos teniendo en cuenta los gastos ni la inversión inicial.

La ocupación alta puede ser beneficiosa al proporcionar una mayor cantidad de días de alquiler, lo que facilita la recuperación rápida de la inversión. Sin embargo, incluso en ciudades con ocupación menor como Barcelona, estrategias de precios y promoción adecuadas pueden maximizar la rentabilidad de los alojamientos.

Al analizar la rentabilidad calculada por nosotros y la ocupación de cada ciudad, se observan relaciones interesantes. Por ejemplo, se destaca la importancia de invertir en la ciudad de Valencia debido a que su rentabilidad no difiere mucho de Madrid y Barcelona, pero supera en ocupación a ambas. Asimismo, se identifica que la rentabilidad más alta corresponde a regiones como Islas Baleares y ciudades como Girona y, sin embargo, sus niveles de ocupación son más bajos. Por lo tanto, es interesante considerar la diversificación de la inversión en sitios como estos para optimizar la rentabilidad global.

Otro punto destacable es que, en el estudio de barrios más rentables dentro de estas 3 ciudades, cuando creamos el dataframe de los barrios más rentables, se observa que diez de estos se encuentran principalmente en Madrid (5) y Valencia (3), lo que resalta la importancia de considerar no solo la ciudad en general, sino también las áreas específicas dentro de cada una de ellas para tomar las decisiones de inversión.

Para concluir, añadir que nuestra labor se basa en el análisis y recopilación de datos adicionales para ayudar a nuestro cliente a tomar decisiones. Si bien proporcionamos esta serie de consejos y recomendaciones basados en los datos disponibles, es importante señalar que la toma de decisiones estratégicas depende de muchos más factores, condicionados por información necesaria de la que no disponemos, para llevarlas a cabo de manera exhaustiva. Se conoce el interés de White Hosting en invertir 300 millones de euros, sin embargo, las decisiones específicas de inversión deben ser evaluadas y adoptadas por el cliente en función de su visión, objetivos y análisis adicionales que pueda realizar.