

PRA 1 ¿Cómo podemos capturar los datos de la web?

Tipología y ciclo de vida de los datos aula 2

Nombres:

- Juan Andrés Dávila
- Ouassim Aouattah

1. Contexto.

Actualmente la oferta de bienes inmuebles es un tema muy interesante de analizar debido a la alta cantidad de ofertas que circulan en la red. Existen varias páginas que ofrecen opciones para que los usuarios puedan buscar ofertas de alquiler o compra. Una de esas opciones es Fotocasa que nos proporciona una interesante cantidad de registros de ofertas de dichos inmuebles para poder obtener información relevante con respecto a las tendencias del mercado.

Enlace al sitio web:

<https://www.fotocasa.es/es/>

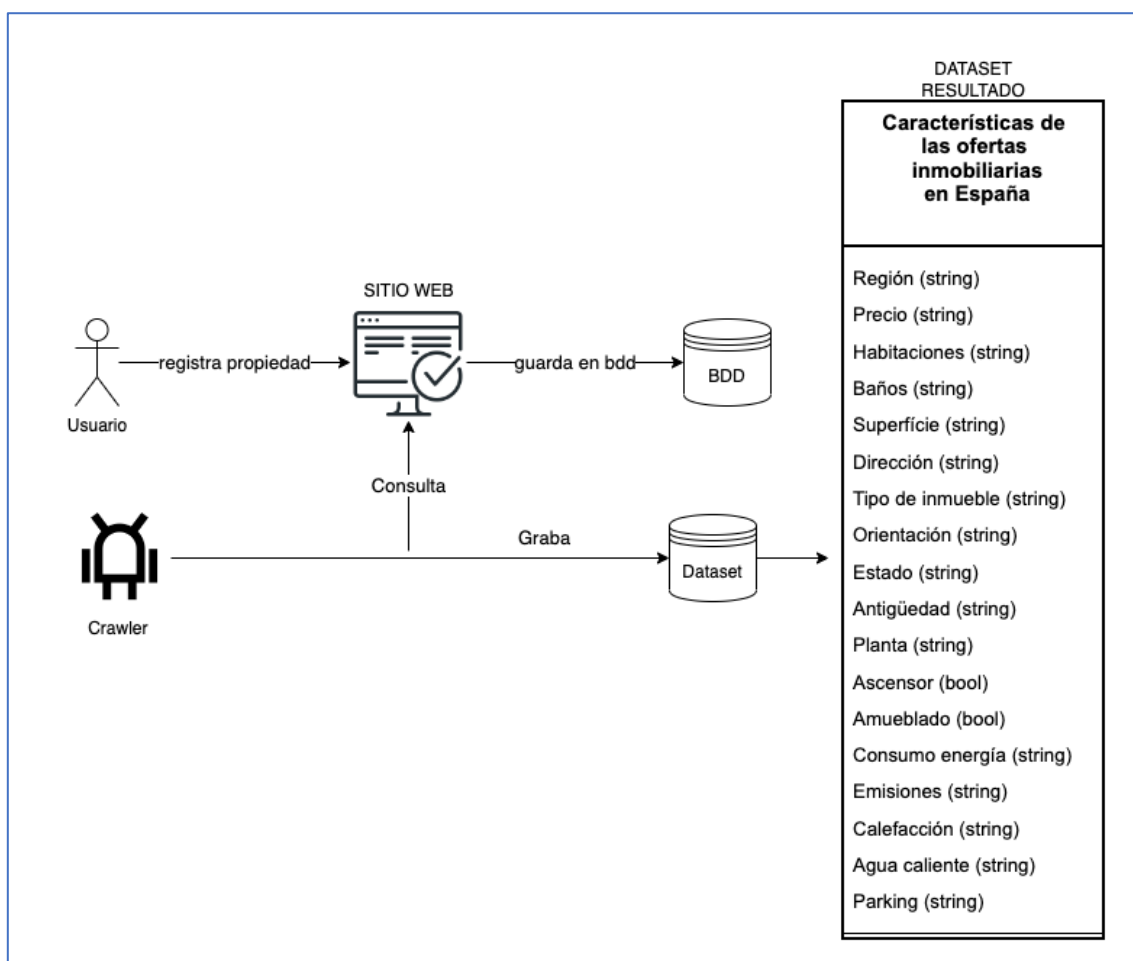
2. Título

Características de las ofertas inmobiliarias en España

3. Descripción del dataset.

El dataset se conforma principalmente de las propiedades o características de las ofertas que los usuarios del sitio web suben o publican para ponerlas a disposición de otros usuarios ya sea para venta o para alquiler. Entre estos atributos esta información valiosa como los precios, tamaño o superficie o el número de habitaciones que nos permitirán obtener más conocimientos sobre el estado del sector inmobiliario en España.

4. Representación gráfica.



5. Contenido.

El dataset se conforma principalmente de las propiedades o características de las ofertas que los usuarios del sitio web suben o postean para ponerlas a disposición de otros usuarios ya sea para venta o para alquiler.

- **Región:** nombre de la región donde se encuentra el inmueble.
Ejemplo: Valencia
- **Precio:** valor de alquiler del inmueble en euros.
Ejemplo: 350.00€
- **Habitaciones:** cantidad de habitaciones.
Ejemplo: 5 habs.
- **Baños:** cantidad de baños.
Ejemplo: 2 baños
- **Superficie:** tamaño en metros cuadrados de la propiedad.
Ejemplo:
- **Dirección:** sector donde se encuentra ubicado el inmueble.
Ejemplo: De Mesonero Romanos - Sol

- **Tipo de inmueble:** tipo de inmueble.
Ejemplo: De Mesonero Romanos - Sol
- **Orientación:** Punto cardinal de la ciudad donde se ubica el inmueble.
Ejemplo: norte
- **Estado:** Describe el estado de mantenimiento del lugar.
Ejemplo: Casi nuevo
- **Antigüedad:** el rango que años que tiene el inmueble desde que se construyó.
Ejemplo: 10 a 20 años
- **Planta:** En qué nivel o planta se encuentra ubicado el lugar.
Ejemplo: 3ª planta
- **Ascensor:** Valor booleano (si o no) que indica si hay ascensor en la propiedad.
Ejemplo: si
- **Amueblado:** Valor booleano (si o no) que indica si la propiedad se encuentra amueblado.
Ejemplo: si
- **Consumo energía:** Indica el consumo energético promedio al año de la propiedad.
Ejemplo: 20 kW h m² / año
- **Emisiones:** Indica la cantidad de emisiones de carbono promedio al año de la propiedad.
Ejemplo: 400 kg CO₂ m² / año
- **Calefacción:** Indica si el inmueble tiene calefacción y si es así de que tipo. Ejemplo: Gas Natural
- **Agua Caliente:** Indica si el inmueble tiene agua caliente y si es así de que tipo. Ejemplo: Electricidad
- **Parking:** Indica si el inmueble tiene parqueadero y si es así de que tipo. Ejemplo: Privado

Los datos que se presentan, como veremos más adelante están siendo invocados de acuerdo con su tiempo de publicación en el sitio web, trayendo en primer lugar las propiedades más recientes en ser publicados.

6. Propietario.

En este caso el propietario de los datos es Adevinta Spain tal como indican en su aviso legal que el usuario del sitio acepta al publicar información en el mismo: *“Al subir fotografías al Portal, el usuario cede gratuitamente a Adevinta Spain los derechos de explotación de propiedad intelectual sobre las mismas, por lo que Adevinta Spain podrá reproducirlas, transformarlas (incluyendo, sin limitación, la inclusión de marcas de agua u otros mecanismos que impidan el aprovechamiento inconsentido por parte de terceros), distribuirlas y comunicarlas al público”* (Adevinta Spain, 2022).

Existe varios estudios similares realizados por el propio sitio sobre los datos que maneja el mismo portal. Por citar un ejemplo tenemos la siguiente información que comunica el portal tras realizar una comparación de los precios de alquiler de sus inmuebles: *“Un hecho que llama la atención ya que, según los datos del Índice Inmobiliario fotocasa se están registrando subidas históricas que no se veían desde 2006 y algunas grandes ciudades han alcanzado los máximos de los años del boom.”* (Fotocasa, 2018)

En este caso de acuerdo al no estar haciendo uso comercial de los datos obtenidos directamente del portal estamos actuando de acuerdo a los éticos y legales, en otras palabras estamos cumpliendo con el anuncio legal de la empresa: *“El Usuario deberá utilizar los materiales, elementos e información a la que acceda a través de la utilización del Portal y de cada uno de los correspondientes Servicios únicamente para sus propias necesidades, obligándose a no realizar ni directa ni indirectamente una explotación comercial ni de los servicios ni de los materiales, elementos e información obtenidos a través de los mismos.”* (Adevinta Spain, 2022).

7. Inspiración

Como se pudo determinar en el apartado anterior el dataset que obtendremos a partir de las ofertas inmobiliarias que alberga el sitio web nos puede ayudar a contestar varias preguntas y no solo referente al tema del precio como se mencionó en la cita. Al tener varios campos como la superficie, consumo energético, antigüedad, orientación, podemos a su vez resolver varias preguntas, como, por ejemplo:

- ¿Qué sector (norte, sur) de una ciudad está más cotizado?
- ¿Cuál es el promedio de antigüedad de una propiedad en una zona de una ciudad?
- ¿Qué tipo de inmueble es el más ofrecido?
- ¿Cuál es el promedio de superficie de un inmueble en una determinada zona?
- ¿Cuál es el promedio de consumo energético en una ciudad?

Estas y muchas otras preguntas nos lleva a considerar que un dataset con información de este sitio es de mucha utilidad para tener un mayor panorama al momento de analizar el sector inmobiliario del país.

8. Licencia

Para el dataset generado en esta práctica hemos usado la licencia “Atribución-NoComercial-CompartirIgual 4.0 Internacional (CC BY-NC-SA 4.0)”

De acuerdo con lo estipulado en dicha ley el dataset se puede compartir y adaptar bajo el termino de no ser usado con fines comerciales. Dar crédito tanto a los creadores y de las partes atribuidas y finalmente si se cambia, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original.

9. Código

- Código de la practica:
https://github.com/XnetLoL/PRA1_WebScrapping
- Librerías utilizadas en la práctica, (ver requeriments.txt):

Nombre	versión
async-generator	1.10
attrs	22.1.0
certifi	2022.9.24
cffi	1.15.1
charset-normalizer	2.1.1
colorama	0.4.6
distlib	0.3.6
filelock	3.8.0
h11	0.14.0
idna	3.4
numpy	1.23.4
outcome	1.2.0
packaging	21.3
pandas	1.5.1
pipenv	2022.11.5
platformdirs	2.5.2
pycparser	2.21
pyparsing	3.0.9
PySocks	1.7.1
python-dateutil	2.8.2
python-dotenv	0.21.0
pytz	2022.6
requests	2.28.1
selenium	4.6.0
six	1.16.0
sniffio	1.3.0
sortedcontainers	2.4.0
tqdm	4.64.1
trio	0.22.0
trio-websocket	0.9.2
urllib3	1.26.12
virtualenv	20.16.6
virtualenv-clone	0.5.7
webdriver-manager	3.8.5
wsproto	1.2.0

- Aspectos relevantes:

- Primero importamos las librerías que se utilizarían para realizar la práctica, en este caso tanto como Selenium y Pandas son las más relevantes tanto para realizar el webscraping como para trabajar con los datos obtenidos respectivamente.

```
# Librerías
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
import time, random
import pandas as pd

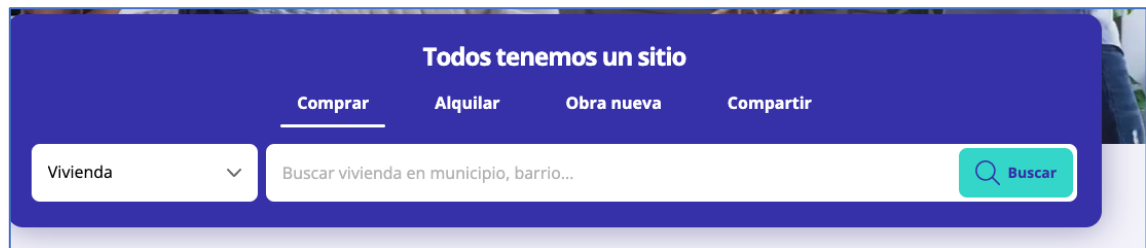
# Opciones de navegación
options = webdriver.ChromeOptions()
options.add_argument('--start-maximized')
options.add_argument('--disable-extensions')
options.add_argument('--profile-directory=Default')
options.add_argument('--ignore-certificate-errors')
options.add_argument('--ignore-ssl-errors')

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
```

- Una vez inicializado el driver seteamos un tiempo muerto mediante la función *sleep* de la librería *time* para evitar la saturación de peticiones y evitar ser bloqueados por el sitio

```
# Inicializamos el navegador
driver.set_window_position(0, 0)
driver.maximize_window()
time.sleep(1)
```

- Creamos una función que permite al usuario buscar propiedades en una determinada zona ya que la página permite buscar una zona



```
def search_region(region):
    # Abrimos la página
    driver.get("https://www.fotocasa.es/es/")
```

- Contralamos un posible pop-up que aparece al abrir, la página.

```
# Cerramos el pop-up
try:
    WebDriverWait(driver, 5)\
        .until(EC.element_to_be_clickable((By.XPATH,
                                            '//*[@id="App"]/div[2]/div/div/div/footer/div/button[2]')))\
        .click()
except:
    pass

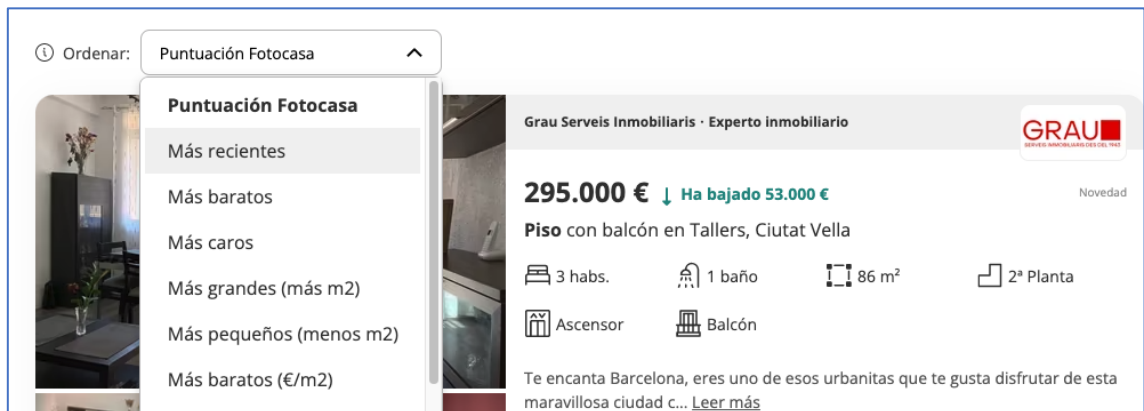
# Buscamos la zona
WebDriverWait(driver, 5)\
    .until(EC.element_to_be_clickable((By.XPATH,
                                        '//*[@id="App"]/div[1]/main/section/div[2]/div/div/div/div/div[2]/div[2]/form/div/div/div/div/div/input')))\
        .send_keys(region)

# Seleccionamos la zona
WebDriverWait(driver, 5)\
    .until(EC.element_to_be_clickable((By.XPATH,
                                        '//*[@id="App"]/div[1]/main/section/div[2]/div/div/div/div/div[2]/div[2]/form/button')))\
        .click()
```

```
# Cerramos el pop-up de alerta
try:
    WebDriverWait(driver, 3)\
        .until(EC.element_to_be_clickable((By.CSS_SELECTOR,
                                                'button.sui-AtomButton.sui-AtomButton--primary.sui-AtomButton--flat.sui-AtomButton--center.sui-AtomButton--fullWidth')))\
        .click()
except:
    pass

# Seleccionamos las opciones de filtro
WebDriverWait(driver, 5)\
    .until(EC.element_to_be_clickable((By.XPATH,
                                        '//*[@id="App"]/div[1]/div[2]/main/div/div[2]/div/div[2]/div/div/input')))\
    .click()
```

- Como mencionamos antes el criterio de búsqueda que utilizamos en esta práctica es la fecha de publicación para ellos abrimos el listado de la página que permite ordenar las publicaciones.



```
# Filtramos por fecha de publicación
WebDriverWait(driver, 5)\
    .until(EC.element_to_be_clickable((By.XPATH,
                                        '//*[@id="App"]/div[1]/div[2]/main/div/div[2]/div/div[2]/div/ul/li[2]')))\
    .click()

# Cerramos el pop-up de alerta
```



```
try:
    WebDriverWait(driver, 3)\
        .until(EC.element_to_be_clickable((By.CSS_SELECTOR,
            'button.sui-AtomButton.sui-AtomButton--primary.sui-AtomButton--flat.sui-
AtomButton--center.sui-AtomButton--fullWidth')))\
        .click()
except:
    pass
```

- Otro punto que vale la pena mencionar es que generamos otra función que permite obtener los datos de las propiedades, indicando la región y el número de registros con el que se desee trabajar.

```
def get_data(region, items):
    # Inicializamos las listas
    rows = items
    df_data = {
        'Región': [region] * rows,
        'Precio': [None] * rows,
        'Habitaciones': [None] * rows,
        'Baños': [None] * rows,
        'Superficie': [None] * rows,
        'Dirección': [None] * rows,
    }
```

- En esta función se encuentra la parte más pesada del código ya que nos encontramos con que el sitio web utiliza una paginación al momento de mostrar resultados, para resolver esto entramos en un bucle que se terminara cuando un contador llegue al número de resultados que se envió como parámetro a la función. En Esta función cargamos los resultados y hacemos scroll para cargar todas las viviendas y obtenemos los enlaces para abrir cada una de las viviendas y extraer los datos, hasta que el contador nos indique cuando parar.

```
i = 0
page = 1

while items > 0:
    time.sleep(random.uniform(1, 3))
```

```
# Navegamos hacia el pie de página para que se carguen todas las viviendas
max = driver.execute_script("return document.body.scrollHeight")
y = 400
while y < max:
    driver.execute_script("window.scrollTo(0, "+str(y)+")")
    y += 400
    time.sleep(random.uniform(0.04, 0.6))

# Obtenemos los enlaces de las viviendas
links = driver.find_elements(By.CSS_SELECTOR, "article>a")
hrefs = [link.get_attribute('href') for link in links]

# Obtenemos los datos de las viviendas
for href in hrefs:
    # Si hemos llegado al límite de items, salimos del bucle
    if items == 0:
        break

    # Abrimos cada vivienda y extraemos los datos
    driver.execute_script("window.open('{}');".format(href))
    driver.switch_to.window(driver.window_handles[1])

    # Esperar un tiempo aleatorio para no ser detectado como bot
    time.sleep(random.randint(1, 2))

    # Navegamos hacia el pie de página para que se carguen todos los datos
    max = 2000
    y = 400
    while y < max:
        driver.execute_script("window.scrollTo(0, "+str(y)+")")
        y += 400
        time.sleep(random.uniform(0.04, 0.5))

    # Extraemos los datos principales
    try:
        df_data['Precio'][i] = driver.find_element(By.CSS_SELECTOR, 'span.re-DetailHeader-price').text
    except:
```

```

pass
try:
    df_data['Habitaciones'][i] = driver.find_element(By.XPATH,
'//*[@id="App"]/div[1]/main/div[3]/div[1]/div/section[1]/div/div[1]/div[3]/ul/li[1]/span[2]').text
except:
    pass
try:
    df_data['Baños'][i] = driver.find_element(By.XPATH,
'//*[@id="App"]/div[1]/main/div[3]/div[1]/div/section[1]/div/div[1]/div[3]/ul/li[2]/span[2]').text
except:
    pass
try:
    df_data['Superficie'][i] = driver.find_element(By.XPATH,
'//*[@id="App"]/div[1]/main/div[3]/div[1]/div/section[1]/div/div[1]/div[3]/ul/li[3]/span[2]').text
except:
    pass
try:
    df_data['Dirección'][i] = driver.find_element(By.XPATH,
'//*[@id="App"]/div[1]/main/div[3]/div[1]/div/section[3]/div/div/div[1]/h2').text.replace(',', ' -')
except:
    pass

labels = driver.find_elements(By.CSS_SELECTOR, 'p.re-DetailFeaturesList-featureLabel')
values = driver.find_elements(By.CSS_SELECTOR, 'p.re-DetailFeaturesList-featureValue')

```

- Los datos que consideramos necesario para la práctica fueron la región, precio, habitaciones, superficie y baños. El sitio muestra opcionalmente varios atributos más que los hemos incluido también y es por eso por lo que el dataset tendrá campos vacíos dependiendo de si la vivienda tiene dicha propiedad o no, como por ejemplo la calefacción

Características



Tipo de inmueble
Piso



Estado
Casi nuevo



Antigüedad
+ 100 años



Planta
3ª planta



Amueblado
No



Consumo energía
E 192 kW h m² / año



Emisiones
E 36 kg CO₂ m² / año

```
# Extraemos los datos secundarios
for label in labels:
    if label.text not in df_data.keys():
        df_data[label.text] = [None] * rows
        df_data[label.text][i] = values[labels.index(label)].text

# Cerramos la pestaña
driver.close()
driver.switch_to.window(driver.window_handles[0])

# Actualizamos el contador
items -= 1
i += 1

# Navegamos a la siguiente página
if items > 0:
    page += 1

    try:
        # Navegamos a la siguiente página
        url = driver.current_url
        next_path = url[:url.find('/')] + '/' + str(page)
        driver.get(next_path)
    except:
        break

return df_data
```

- Finalmente utilizamos las dos funciones creadas anteriormente y con la librería de pandas creamos un dataframe en formato csv

```
def scrap(regions, items):
    # Creamos los dataframes
    df = pd.DataFrame()
    try:
        for region in regions:
```

```
# Navegamos a la región
search_region(region)

# Obtenemos los datos
df_data = get_data(region, items)

# Añadimos los datos al dataframe
df = df.append(pd.DataFrame(df_data), ignore_index=True)

except:
    pass

return df
```

- Aquí en el ejemplo buscamos en 12 regiones y con 50 resultados por región

```
if __name__ == '__main__':
    # Regiones a buscar
    regions = ['Madrid', 'Barcelona', 'Valencia', 'Málaga', 'Sevilla', 'Zaragoza', 'Bilbao', 'Mallorca', 'Tenerife',
'Gran Canaria']

    # Profundidad de la búsqueda
    items = 50

    # Realizamos la búsqueda
    df = scrap(regions, items)

    # Guardamos los datos en un csv
    df.to_csv('data.csv', index=False).
```

- Tal como indicaba la practica comprobamos que User Agent estamos utilizando, dándonos como resultado en este caso:
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36

```
# Imprimimos el User-Agent
print(driver.execute_script("return navigator.userAgent"))

# Cerramos el navegador
driver.quit()
```

10. Dataset








- Dataset publicado en Zenodo (DOI 7343529):
<https://zenodo.org/record/7343529>

11. Vídeo

Enlace al video:

https://drive.google.com/file/d/1kFePAPDdWmU6hC8SFsw_IV6E5hFaNVSU/view?usp=share_link

12. Contribuciones:

Contribuciones	Firmas	
Investigación previa	Integrante 1: 	Integrante 2: 
Redacción de las respuestas	Integrante 1: 	Integrante 2: 
Desarrollo del código	Integrante 1: 	Integrante 2: 
Participación en el vídeo	Integrante 1: 	Integrante 2: 