***Alumno**: Anton Barrera Mora (abarreramoraùoc.edu).*

¿Cómo podemos capturar los datos de la web?

Sitio web escogido para web scraping:
https://www.pricecharting.com/es/

Dataset en Zenodo:
https://doi.org/10.5281/zenodo.14043146

Repositorio en GitHub:
https://github.com/Kamaranis/Web-Scraping-de-videojuegos-con-potencial-de-revalorizacion.git

Invitación para *jmoreiras-uoc* GitHub:
https://github.com/Kamaranis/Web-Scraping-de-videojuegos-con-potencial-de-revalorizacion/invitations

Video presentación en Google Drive de la UOC:
https://drive.google.com/file/d/1PB7gAi-zFeUey35RGmEGJwlCjwKJDR8E/view?usp=drive_link

En caso de que presente problemas, adjunto enlace a la carpeta donde de Google Drive donde esta contenido el video (se ha concedido permisos a jmoreiras@uoc.edu):

https://drive.google.com/drive/folders/1MbdzbXIaVSWZA0eO7XRL1pcy78sTT_7Q?usp=drive_link

Indice

- ¿Cómo podemos capturar los datos de la web? - Indice
  - 1. Contexto
    - VGChartz
    - PriceCharting
      - Conclusiones del Análisis Preliminar de Pricecharting.com para Web Scraping
  - 2. Título
  - 3. Descripción del dataset
  - 4. Representación gráfica
  - 5. Contenido
  - 6. Propietario
  - 7. Inspiración
  - 8. Licencia
    - Dataset
    - Código Fuente
  - 9. Código
  - 10. Dataset
  - 11. Video
  - Bibliografía

1. Contexto

En este apartado describiremos de forma general, el proceso seguido para desarrollar el presente trabajo de Web Scraping para la PR1.

Este proyecto nació con un enfoque lúdico, orientado a realizar **web scraping** en sitios web del ámbito del entretenimiento (videojuegos, juegos de mesa, libros y cartas coleccionables). Esta elección temática responde a un interés personal por explorar áreas más recreativas, como contrapunto a la actividad profesional cotidiana en el sector IT.

La selección del tema para la "Práctica 1" (PR1) de la asignatura **Tipología y Ciclo de Vida de los Datos** se realizó considerando tres criterios fundamentales:

1. El área de interés personal.
2. La disponibilidad y accesibilidad de datos en sitios web.
3. La viabilidad para realizar un análisis significativo que cumpliera con los requisitos de la práctica.

Todo el proceso se ha desarrollado prestando especial atención a los aspectos técnicos y éticos inherentes al web scraping, garantizando así un trabajo riguroso y profesional.

El proceso de trabajo puede consultarse íntegramente en el documento incluido en el repositorio de la práctica, un notebook de Google Colab.

En este documento puede observarse que el primer paso fue **obtener un listado de sitios web** relacionados con la temática seleccionada.
Trabajamos sobre estos sitios web inicialmente para conocer que posibilidades y limitaciones ofrecian:

1. **SteamDB**
   Base de datos de la plataforma Steam. Aunque la API de SteamDb limita parte de los datos accesibles, es posible obtener información sobre los videojuegos disponibles. Cuenta con API de pago.

2. **BoardGameGeek**
   Información sobre juegos de mesa, con datos sobre la puntuación, complejidad y la mecánica de los juegos, entre otros aspectos.

3. **Metacritic Video Games**
   Recopilación de reseñas de videojuegos, donde se asigna una puntuación media tanto por parte de la crítica especializada como de los usuarios. Incluye datos como el nombre del juego, la plataforma, la fecha de lanzamiento y el género.

4. **Howlongtobeat**
   Proporciona estimaciones sobre la duración de los videojuegos, tanto para completar la historia principal como para lograr un 100% de finalización.

5. **VGChartz**
   Información sobre ventas de videojuegos, incluyendo el número de unidades vendidas por plataforma y región.

6. **PriceCharting**
   Rastrea el precio histórico de videojuegos, consolas, juegos de mesa y cartas coleccionables, facilitando un análisis sobre la evolución de los precios a lo largo del tiempo. Cuenta con una API y funciones de pago que no parecen interferir en el Web Scraping.

Cabe destacar que se trataba de sitios web en inglés a excepción de www.pricecharting.com que cuenta con traducción a diversos idiomas, incluido el español.

Posteriormente se procedió a una segunda fase que denominamos **evaluación preliminar** de la viabilidad del scraping en estos sitios. El objetivo era valorar las restricciones técnicas, legales y la accesibilidad de los datos mediante un **análisis de los archivos** `robots.txt` cuidadoso con base en el listado completo anteriormente referenciado.

Como resultado de este análisis del archivo robots.txt del sitio web finalmente escogido para esta práctica (PriceCharting.com) concluimos que esta web bloquea el acceso a secciones relacionadas con la gestión de cuentas, la publicación de ofertas y la compra de productos. Se tratan de secciones del sitio web dinámicas y personalizadas para cada usuario. Para el web scraping de información de precios, el archivo robots.txt no presentaba restricciones. En un principio todo parecía indicar que se podría acceder a las páginas de productos y extraer la información, siempre de forma responsable y ética.

El resto de sitios web a excepción del 'robots.txt' de VGchartz presentaban restricciones de diversa índole que llevaron a sus descartes en esta fase. Consecuentemente, procedimos a una tercera fase de **Análisis preliminar** con solo dos candidatos: "pricecharting.com" y "vgchartz.com", respectivamente. Análisis preliminar que desglosamos a continuación:

VGChartz

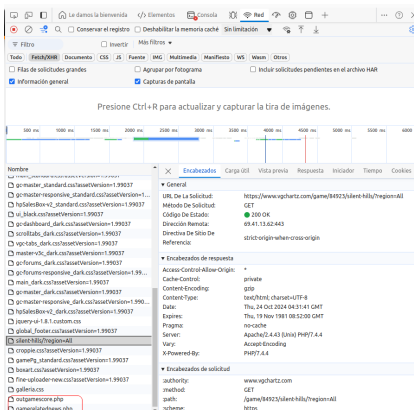
Se efectuó un análisis preliminar del sitio web como puede observarse en el documento de trabajo que tuvo como objetivo evaluar la factibilidad y los desafíos de realizar web scraping en el sitio web VGChartz (www.vgchartz.com) empleando diversas herramientas y técnicas de análisis. Finalmente descartamos este sitio, pero a modo de resumen, detallamos los hallazgos:

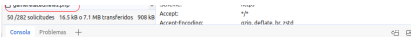
Análisis de tecnologías:

Mediante la librería `builtwith` de Python, se identificó que VGChartz utiliza `PHP` como lenguaje de programación principal, lo que conlleva generación dinámica de contenido. Esto implica que la información de interés no estará embebida directamente en el código HTML, sino que se obtendrá a través de peticiones a una base de datos subyacente. La presencia de `jQuery` como framework de `JavaScript` refuerza esta hipótesis, ya que facilita la implementación de peticiones AJAX para la carga dinámica de contenido.

Inspección de la actividad de red:

Un análisis de la actividad de red con las herramientas de desarrollo del navegador, específicamente la pestaña "Red" con el filtro "XHR/Fetch" activado, confirma la naturaleza dinámica del sitio web. Se observan múltiples peticiones asíncronas a archivos PHP, como `outgamescore.php` y `gamerelatednews.php`, que sugieren la interacción con la base de datos para obtener y renderizar la información en la página.





Vista de las 'DEV Tolls' en el sitio vgchartz.com.

Tamaño del sitio web:

El análisis del sitio web nos permite vislumbrar que este es un sitio web masivo en cuanto a numero de paginas, 175.000 lo que supone un reto añadido al reto que supone en sí mismo el stack empleado.

Análisis del archivo robots.txt:

El archivo robots.txt de VGChartz se muestra permisivo, permitiendo el acceso a todas las secciones del sitio web. Sin embargo, esto no descarta la posible implementación de otras medidas anti-scraping, como la detección de patrones de acceso o el bloqueo de IPs.

Información WHOIS:

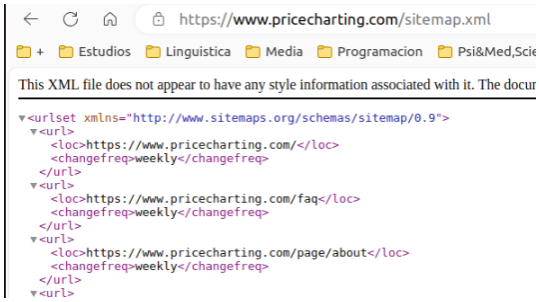
La consulta WHOIS revela que el dominio VGChartz se registró en 2007, lo que sugiere una larga trayectoria y una posible complejidad en la estructura del código. Si bien esta no aporta detalles técnicos relevantes para el web scraping, contextualiza la antigüedad del sitio web y su gestión.

Consideraciones para el web scraping:

- **Contenido dinámico:** La utilización de `PHP` y `jQuery` implica la necesidad de emplear técnicas de web scraping que manejen la renderización de JavaScript y la extracción de datos de peticiones AJAX. Herramientas como *Scrapy* y análogas, con capacidad para procesar JavaScript y simular la interacción del usuario, serian inevitables en este trabajo.
- **Posibles medidas anti-scraping:** A pesar de la permisividad del archivo 'robots.txt', se debe considerar la posibilidad de encontrar medidas anti-scraping. Será necesario monitorear el comportamiento del sitio durante el proceso y adaptar la estrategia en consecuencia.
- **Inspección del código fuente:** Un análisis detallado del código fuente de las páginas sería fundamental para comprender la estructura de la información y la forma en que se cargan los datos dinámicos. Esto permitiría identificar los selectores `CSS` o `XPath` empleados en la extracción de datos dependiendo de la librería a emplear, énfasis con `Scrapy`, aunque finalmente hemos descartado trabajar sobre este sitio web. Principalmente porque a priori parece haber alternativas con menos complicaciones y mejores fuentes de datos como es el caso de Pricecharting.com.

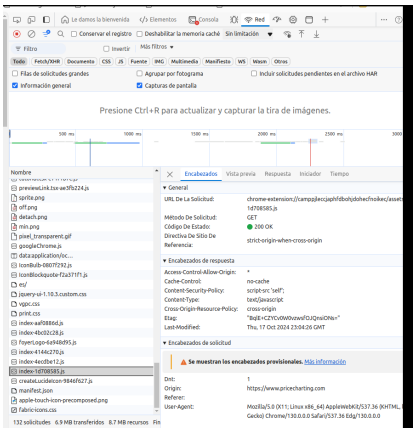
PriceCharting

PRICE charting tiene activo el sitemap que permitiría afinar el código para realizar el scraping solo sobre las páginas relevantes para el proyecto, como puede observarse en la imagen:



Vista del código XML del sitemap de Pricecharting.com.

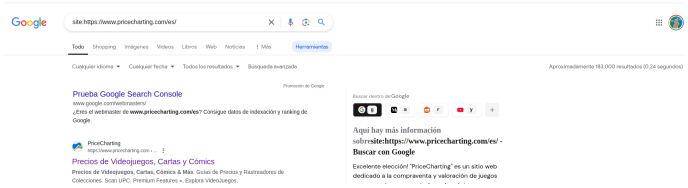
La inspección de Pricecharting con las herramientas de desarrollo del navegador (Dev Tools) revelan un comportamiento distinto al observado en VGChartz en cuanto a la carga de la información de precios.



Vista de las 'Dev Tools' en PriceCharting.com.

El análisis con las 'Dev Tools' revela que muy posiblemente el sitio utilice JavaScript embebido. La información de precios podría estar presente en el código HTML, ya sea incrustada en etiquetas HTML o como atributos de datos en elementos HTML. El JavaScript de la página se encargaría de procesar y mostrar la información sin necesidad de realizar peticiones adicionales al servidor.

Respecto al tamaño, empleando 'site:https://www.pricecharting.com/es/' en el buscador y como puede observarse, obtenemos que este sitio web tiene aproximadamente, **183.000 paginas**.



Estimación del tamaño del sitio web pricecharting.com usando Google.com.

Asimismo, para obtener información sobre las tecnologías utilizadas en https://www.pricecharting.com/es/ , empleamos la librería de Python `builtwith`:

```
import builtwith
import pprint

# Creamos un diccionario para guardar los resultados
informacion_sitio = builtwith.parse('https://www.pricecharting.com/')

# Imprimimos la información con pprint
pprint.pprint(informacion_sitio)

# RESULTADOS:

{'javascript-frameworks': ['jQuery', 'jQuery UI'],
 'web-servers': ['Google App Engine']}
```

Pricecharting está alojado en Google App Engine y utiliza tecnologías Javascript para mostrar contenidos sin necesidad de recargar la pagina.
Esto tiene varias connotaciones para el web scraping, algunas positivas y otras que podrían presentar desafíos.

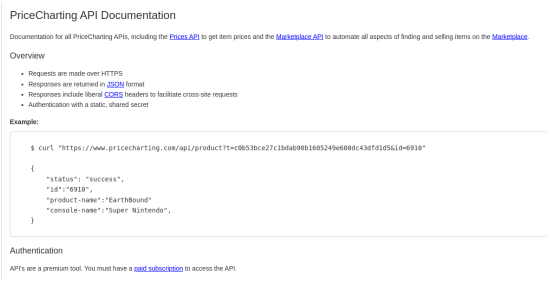
Posibles ventajas:

- Escalabilidad y rendimiento: Google App Engine está diseñado para escalar automáticamente en función del tráfico, lo que significa que el sitio web debería ser capaz de manejar un gran número de peticiones sin problemas de rendimiento.
- Infraestructura robusta: Google App Engine ofrece una infraestructura seria de alta disponibilidad y protección contra ataques DDoS. Ello conlleva implícitamente que el sitio web debería estar accesible sin problemas, lo que facilitaría web scraping continuo.
- Posible uso de APIs: Google App Engine facilita la creación y el despliegue de APIs. Pricecharting ofrece una API para acceder a los datos pero que afectos de esta practica, no podremos utilizar

Probables desafíos:

- Protección contra web scraping: Google App Engine ofrece herramientas y servicios para proteger las aplicaciones contra el web scraping, como la detección de patrones de acceso, el bloqueo de IPs y la integración con reCAPTCHA. No cabe duda de que Pricecharting podría estar utilizando algunas de estas medidas, lo que dificultaría el trabajo.

- Limitaciones de acceso: Google App Engine puede imponer limitaciones en el número de peticiones que se pueden realizar a una aplicación, especialmente teniendo en cuenta que tiene API. Al poseer una API de pago, es muy probable que el intento de web scrap finalice en resultados limitados, como puede observarse en la captura al pie.


PriceCharting API Documentation

Documentation for all PriceCharting APIs, including the Prices API to get item prices and the Marketplace API to automate all aspects of finding and selling items on the Marketplace.

Overview

- Requests are made over HTTPS
- Responses are returned in JSON format
- Responses include liberal CORS headers to facilitate cross-site requests
- Authentication with a static, shared secret

Example:

```
$ curl "https://www.pricecharting.com/api/product?t=c0b53bce27c1bdab90b1605249e600dc43dfd1d56id=6910"

{
    "status": "success",
    "id":"6910",
    "product-name":"EarthBound"
    "console-name":"Super Nintendo",
}
```

Authentication

APIs are a premium tool. You must have a paid subscription to access the API.

Limitaciones de acceso a la API de Pricecharting.

- Cambios en la infraestructura: Google suele implementar cambios en las condiciones e infraestructuras, afectando al Web Scraping. Aunque solo necesitamos realizar puntualmente Scraping para esta practica, en un contexto laboral, en un futuro, el diseño del Scraper podría quedar inservible.

Vamos a realizar un `whois` para conocer al propietario y otros detalles, de figurar esta información:

```
import whois

# Realizamos el query
pprint.pprint(whois.whois('https://www.pricecharting.com/'))

RESULTADOS:

{'address': 'REDACTED FOR PRIVACY',
 'city': 'REDACTED FOR PRIVACY',
 'country': 'US',
 'creation_date': datetime.datetime(2010, 7, 23, 19, 18, 48),
 'dnssec': 'unsigned',
 'domain_name': ['PRICECHARTING.COM', 'pricecharting.com'],
 'emails': 'abuse-complaints@squarespace.com',
 'expiration_date': datetime.datetime(2025, 7, 23, 19, 18, 48),
 'name': 'REDACTED FOR PRIVACY',
```

```
  'name_servers': ['NS-CLOUD-B1.GOOGLEDOMAINS.COM',
                   'NS-CLOUD-B2.GOOGLEDOMAINS.COM',
                   'NS-CLOUD-B3.GOOGLEDOMAINS.COM',
                   'NS-CLOUD-B4.GOOGLEDOMAINS.COM',
                   'ns-cloud-b3.googledomains.com',
                   'ns-cloud-b4.googledomains.com',
                   'ns-cloud-b1.googledomains.com',
                   'ns-cloud-b2.googledomains.com'],
  'org': 'Video Game Price Charts, LLC',
  'referral_url': None,
  'registrant_postal_code': 'REDACTED FOR PRIVACY',
  'registrar': 'Squarespace Domains II LLC',
  'state': 'CO',
  'status': ['clientDeleteProhibited '
             'https://icann.org/epp#clientDeleteProhibited',
             'clientTransferProhibited '
             'https://icann.org/epp#clientTransferProhibited',
             'clientTransferProhibited '
             'http://www.icann.org/epp#clientTransferProhibited',
             'clientDeleteProhibited '
             'http://www.icann.org/epp#clientDeleteProhibited'],
  'updated_date': [datetime.datetime(2024, 7, 8, 22, 46, 11),
                   datetime.datetime(2024, 7, 8, 22, 46, 11, 510181)],
  'whois_server': 'whois.squarespace.domains'}
```

La consulta WHOIS realizada sobre el dominio pricecharting.com, utilizando la librería whois de Python, proporciona información relevante sobre el sitio web, aunque no aporta detalles técnicos que tengan una importancia diferencial en la estrategia de web scraping.

Información relevante:

- Fecha de creación: El dominio se registró en julio de 2010. Ello supone una presencia establecida en la web. Si bien no es un factor determinante, la antigüedad del sitio web podría sugerir una mayor complejidad en su estructura y la posible implementación de medidas anti-scraping a lo largo del tiempo.
- Fecha de expiración: El dominio expira en julio de 2025, confirmando la continuidad de las operaciones del sitio. Organización: El dominio está registrado a nombre de "Video Game - Price Charts, LLC", una entidad.
- Registrador: El registrador del dominio es '*Squarespace Domains II LLC*'.
- Servidores de nombres: Servidores de nombres de "Google Domains", lo que podría estar relacionado con el alojamiento del sitio web en Google App Engine.
- Estado del dominio: Implementa medidas de seguridad comunes para proteger la propiedad del dominio.

Implicaciones para el web scraping:

Aunque la información reseñada no proporciona detalles técnicos sobre la tecnología del sitio web, algunos aspectos pueden ser relevantes:

- Antigüedad: La trayectoria de Pricecharting sugiere la posibilidad de que se hayan implementado medidas anti-scraping a lo largo del tiempo.
- Infraestructura del servidor: El uso de servidores de nombres de Google Domains indica una infraestructura robusta y escalable, lo influye en la velocidad y capacidad de respuesta del servidor y manejar múltiples peticiones HTTPS.

Realizamos un acercamiento de prueba al Web Scraping:

```
import requests
from bs4 import BeautifulSoup

url: str = ("https://www.pricecharting.com/es/category/video-games")
sitio = requests.get(url)
print("Codigo HTTP:", sitio.status_code)
print("El contenido de la pagina en bruto:")
# pprint.pprint(sitio.content)
# Evito usar esta ultima linea por su verbosidad

RESULTADO:

Codigo HTTP: 200
```

El código 200 nos indica que se ha producido la respuesta a la petición HTTP.
Continuamos analizando el contenido de la página:

```
# Parseamos el contenido con 'BeatifulSop'
price_chart_parsed = BeautifulSoup(sitio.content)
# Imprimimos el contenido de forma legible con la función `prettify`:
print(price_chart_parsed.prettify())
```

Observamos que la subpágina de Pricecharting.com referida a video juegos en Español nos proporciona los enlaces a los videojuegos clasificados por sistemas como puede observarse en el código HTML:

```
<ul>
        <li>
         <a href="/es/console/xbox" title="Xbox">
          Xbox
         </a>
        </li>
        <li>
         <a href="/es/console/sega-genesis" title="Sega Genesis">
          Sega Genesis
         </a>
        </li>
        <li>
         <a href="/es/console/xbox-one" title="Xbox One">
          Xbox One
         </a>
        </li>
        [...]
</ul>
```

En caso de raspar una página concreta como ejemplo empleando un enlace a videojuegos de 'PSOne' (https://www.pricecharting.com/es/console/playstation), observaremos que los precios están referenciados en el mismo código HTML:

```
import requests
from bs4 import BeautifulSoup
url: str = ("https://www.pricecharting.com/es/console/playstation")
sitio = requests.get(url)
# Parseamos el contenido con 'BeatifulSop'
price_chart_parsed = BeautifulSoup(sitio.content)
# Imprimimos el contenido de forma legible con la función `prettify`:
print(price_chart_parsed.prettify())

# RESULTADOS:
[...]
<td class="title" title="4280">
           <a href="/es/game/playstation/breath-of-fire-3">
            Breath of Fire 3
           </a>
          </td>
          <td class="price numeric used_price">
           <span class="js-price">
            $43.24
           </span>
          </td>
          <td class="price numeric cib_price">
           <span class="js-price">
            $68.84
           </span>
          </td>
          <td class="price numeric new_price">
           <span class="js-price">
            $322.82
           </span>
          </td>
[...]
```

Pero asimismo detectamos un problema adicional, pues como es lógico, el sitio web implementa técnicas de "*lazy load*" y carga los resultados progresivamente a medida que el usuario hace *scroll* en la página que muestra en los resultados utilizando un script como puede observarse en el código:

```
 <form method="POST" action="" class="next_page js-next-page">
                       <input type="hidden" name="sort" value="" />
```

Este fragmento de código revela que la página utiliza JavaScript para cargar contenido dinámicamente a medida que el usuario interactúa con ella (en este caso, al hacer scroll). Esto implica que, si simplemente descargamos el HTML inicial con `urllib` o similares, solo obtendremos una parte de los datos, ya que el resto se carga posteriormente mediante JavaScript.
Para solucionar esto, deberemos utilizar otras estrategias para afrontar la carga dinámica del sitio web.

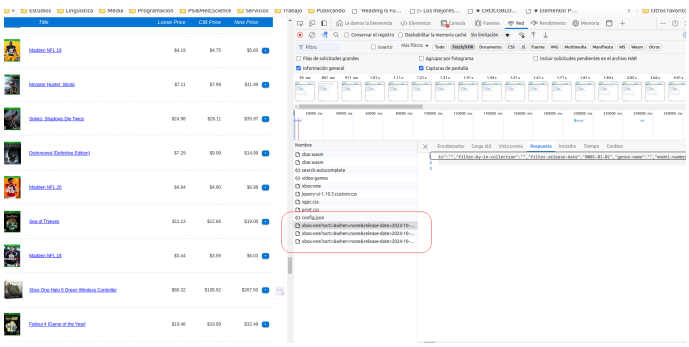
Conclusiones del Análisis Preliminar de Pricecharting.com para Web Scraping

El análisis preliminar llevado a cabo con el objetivo de evaluar la factibilidad y los desafíos del web scraping, ha desvelado información clave sobre la arquitectura del sitio web y las tecnologías que lo componen. Los principales hallazgos y sus implicaciones para el proceso de *web scraping* podrían resumirse en:

Hallazgos principales:

- **Contenido dinámico y "Lazy Load":** Pricecharting.com utiliza JavaScript para generar contenido dinámico, incluyendo la información de precios. Además, implementa "*lazy load*", cargando los datos de los videojuegos a medida que el usuario hace scroll en la página. Esto requiere una solución de *web scraping* que pueda ejecutar JavaScript y simular la interacción del usuario para obtener la información completa.

- **Peticiones XHR para la carga dinámica:** Se observa que la página utiliza peticiones XHR para cargar la información de precios de forma dinámica al hacer scroll. Aunque los datos se muestran en el HTML, la carga de nuevos productos se realiza mediante peticiones XHR que se envían al servidor a medida que el usuario interactúa con la página.



Estructura de las peticiones XHR para carga de datos en pricecharting.com.

- **Alojamiento en Google App Engine:** El sitio web está alojado en Google App Engine, lo que puede implicar la presencia de medidas anti-scraping y limitaciones de acceso, aunque en las pruebas realizadas no hayan surgido problemas técnicos.

- **API comercial:** Pricecharting.com ofrece una API comercial, pero queda descartada para este proyecto debido a restricciones del ejercicio.

- **Sitemap disponible:** El sitio web proporciona un sitemap, que podría ser útil para identificar las URLs principales, pero no para la extracción completa de datos debido al "*lazy load*".

Implicaciones para el Web Scraping:

- **Carga progresiva de datos:** Se requiere una herramienta o estrategia que permita simular el comportamiento del navegador con el objetivo de cargar todos los datos.
- **Análisis del código fuente:** El análisis del código fuente ha permitido identificar la estructura de la información y la lógica del uso del código JavaScript.
- **Consideración de medidas anti-scraping:** La presencia de medidas anti-scraping y uso de técnicas para evitar el bloqueo, como proxies y rotación de user-agents no queda descartado
- **Respeto a las limitaciones de acceso:** Es fundamental respetar las limitaciones de acceso y utilizar técnicas de web scraping éticas y responsables.

Finalizamos este apartado reseñando que con toda la información analizada, **nos decantamos por realizar Web Scrap sobre el sitio web www.pricecharting.com** con base en que los datos que ofrece son relevantes para

realizar diferentes trabajos de valoración de videojuegos, evolución de precios y los aspectos tanto técnicos como éticos son abordables.

2. Título

Web Scraping de Videojuegos con potencial de revalorización

3. Descripción del dataset

El dataset generado consiste en un conjunto de datos estructurados que recopilan información sobre videojuegos disponibles para las consolas *Xbox One* y *Xbox Series X*. Este dataset se construyó a partir de la extracción de datos (web scraping) del sitio web Pricecharting.com, utilizando un script desarrollado en Python.

Características principales del dataset:

- Estructura: Se organiza en un formato tabular. Cada fila representa un videojuego y cada columna una característica.
- Tamaño: El dataset contiene un total de 2369 filas o registros y 6 columnas.
- Fuente: Los datos se obtuvieron exclusivamente del sitio web Pricecharting.com.
- Fecha de extracción para archivos parciales: La fecha de extracción de los datos se registra en el nombre del archivo CSV generado (AAAA-MM-DD) en archivos parciales dependiendo de la plataforma, aunque el archivo final no la lleva impresa.
- La fecha de extracción es **2024/11/05**.

Usos del dataset:

Para nuestro "*Análisis de la revalorización de videojuegos mediante web scraping*", el dataset desempeñará el siguiente papel:

- Identificar videojuegos con potencial de revalorización en las plataformas Xbox One y Xbox Series X.
- Analizar la evolución de los precios de los videojuegos desde su lanzamiento hasta la actualidad (año 2024).
- Calcular el porcentaje de revalorización de cada videojuego, considerando el precio de lanzamiento ("Precio Nuevo") y los precios actuales ("Precio Suelto" y "Precio CIB").
- Ayudar eventualmente a identificar los factores que influyen en la revalorización de los videojuegos, como la popularidad, la crítica y la escasez.
- Visualizar los resultados mediante gráficos y tablas que permitan identificar patrones y tendencias en la revalorización de los videojuegos.

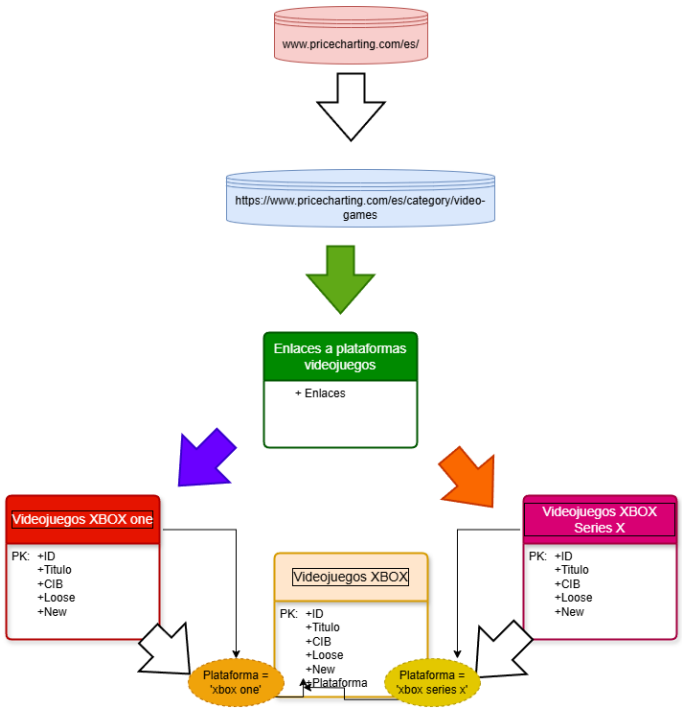
Metodología:

- Se realizó web scraping del sitio web Pricecharting.com para obtener primero los enlaces relevantes y luego se capturó información sobre los precios de los videojuegos.
- Se empleó la librería `Requests-HTML` de Python para extraer los datos.
- El porcentaje de revalorización se calculará como la diferencia entre el precio actual y el precio de lanzamiento, dividida entre el precio de lanzamiento.
- Se crearán gráficos de líneas para visualizar la evolución de los precios a lo largo del tiempo, y gráficos de barras para comparar la revalorización entre diferentes juegos.

Limitaciones:

- **Representatividad:** El dataset solo incluye videojuegos listados en Pricecharting.com, por lo que puede no ser completamente representativo del mercado de videojuegos en su totalidad.
- **Sesgo geográfico:** Los precios en Pricecharting.com pueden estar influenciados por la ubicación geográfica de los usuarios y vendedores, lo que podría introducir un sesgo en los datos.
- **Actualización:** Los precios de los videojuegos son dinámicos, por lo que el dataset requiere actualizaciones periódicas para mantener su validez.

4. Representación gráfica



Representación gráfica Diagrama Entidad Relación dataset videojuegos.

5. Contenido

- Variables: Las variables incluidas en el dataset son:
  - ID: Identificador único del videojuego en la plataforma web.
  - Título: Nombre del videojuego.
  - Loose: Precio del videojuego en estado "loose" (sin caja original).
  - CIB: Precio del videojuego "complete in box" (con caja original).
  - New: Precio del videojuego nuevo.
  - Plataforma: Consola para la que está disponible el videojuego (Xbox One o Xbox Series X). **El script puede adaptarse para descargar datos de otras plataformas.**

6. Propietario

El propietario de los datos extraídos en este trabajo es Pricecharting.com, una plataforma web que proporciona información sobre precios de videojuegos.
El dominio Pricecharting.com está registrado a nombre de *Squarespace Domains II LLC*.

Para asegurar el cumplimiento de los principios éticos y legales en este proyecto, se han realizado las siguientes acciones:

- Análisis de robots.txt: Se ha revisado el archivo robots.txt de Pricecharting.com para identificar posibles restricciones al acceso a la información. El análisis reveló que no existen limitaciones para la extracción de

datos de precios de videojuegos.

- Revisión de los términos de uso: Se han consultado los términos de uso de Pricecharting.com para verificar que la actividad de scraping realizada en este trabajo no infringe ninguna de sus cláusulas. Los términos de uso permiten la extracción de datos para fines no comerciales, siempre que se realice de forma responsable y sin sobrecargar el servidor.
- Limitación de la frecuencia de peticiones: Para evitar la sobrecarga del servidor, el script de scraping implementa un retraso de X segundos entre cada petición siempre que ello ha sido necesario.
- Uso académico de los datos: Los datos extraídos se utilizarán exclusivamente para fines académicos en el contexto de la asignatura "Tipología y ciclo de vida de los datos". No se prevé ningún uso comercial de los datos.

La extracción de datos de Pricecharting.com se justifica por la necesidad de obtener información actualizada y precisa sobre los precios de los videojuegos para las consolas Xbox One y Xbox Series X. Esta información es fundamental para el análisis de precios y tendencias en el mercado de videojuegos, y no está disponible de forma tan completa y estructurada en otras fuentes.

7. Inspiración

Este conjunto de datos puede resultar de gran interés para diversos análisis e investigaciones en el ámbito de los videojuegos y la economía. Su valor radica en la información detallada que proporciona sobre los precios de los videojuegos en diferentes estados de conservación (loose, CIB, new) y plataformas (Xbox One y Xbox Series X).

A partir de este conjunto de datos, se pueden plantear las siguientes preguntas de investigación:

- **¿Existe una diferencia significativa en los precios de los videojuegos entre las plataformas Xbox One y Xbox Series X?** Se podría analizar si los juegos más recientes o con mayor demanda tienen un precio mayor en la plataforma más nueva.
- **¿Cómo varían los precios de los videojuegos en función de su estado de conservación?** Se podría comparar la diferencia de precio entre juegos loose, CIB y new, y determinar si existen patrones o factores que influyen en estas variaciones.
- **¿ Podemos identificar joyas o futuros juegos revalorizables?** ¿Se podría establecer o listar videojuegos que sean susceptibles de revalorización?
- **¿Se puede predecir la evolución futura de los precios de los videojuegos a partir de los datos históricos?** Se podrían desarrollar modelos predictivos para estimar el valor futuro de los juegos.

Comparación con análisis anteriores:

Aunque no se disponen de análisis previos específicos sobre este conjunto de datos, existen estudios similares en el ámbito de la economía de los videojuegos que han abordado algunas de estas preguntas. Por ejemplo, investigaciones previas han analizado la influencia de la popularidad, la crítica y la escasez en los precios de los videojuegos retro (consolas antiguas) (Goh et al., 2023; Writer, 2023). Este conjunto de datos permitiría realizar análisis para consolas modernas como *Xbox One* y *Xbox Series X*, y servir como guía.

Además, este conjunto de datos ofrece la posibilidad de realizar análisis novedosos, como la comparación de precios entre diferentes estados de conservación (loose, CIB, new) y la predicción de la evolución futura de los

precios. Estos análisis podrían aportar información valiosa para coleccionistas, vendedores y compradores de videojuegos, así como para investigadores interesados en la economía de los videojuegos.

8. Licencia

En este proyecto, se han considerado cuidadosamente las licencias para el dataset y el código fuente, con el objetivo de promover el acceso abierto a la información y al mismo tiempo respetar los derechos de los creadores de los datos.

Dataset

El dataset resultante de este proyecto se publicará bajo la licencia CC0: Public Domain License. Esta licencia renuncia a todos los derechos de autor y coloca los datos en el dominio público, permitiendo su uso, modificación y distribución sin restricciones, incluso para fines comerciales. La elección de CC0 se basa en los siguientes principios:

- Reconocimiento del origen de los datos: Los datos extraídos provienen de Pricecharting.com, un sitio web que ofrece información pública sobre precios de videojuegos.
  Al liberar el dataset al dominio público, se reconoce que los datos originales no son propiedad del autor de este proyecto.
- Promoción del acceso abierto: CC0 facilita el acceso y la reutilización de los datos, fomentando la investigación, el análisis y la creación de nuevas aplicaciones.
- Limitación de la extracción de datos: Como Pricecharting.com ofrece una API premium para acceder a sus datos, se ha limitado la cantidad de información extraída para este proyecto, como muestra de respeto a su modelo de negocio.

Código Fuente

El código fuente del proyecto se publicará bajo la licencia CC BY-NC-SA 4.0. Esta licencia permite la reutilización y modificación del código, siempre que se atribuya el crédito al autor original, no se utilice para fines comerciales y se compartan las modificaciones bajo la misma licencia. Esta elección se justifica por el deseo de proteger la autoría del código y al mismo tiempo permitir su uso en contextos académicos y no comerciales.

9. Código

El código resultante para realizar *Web Scraping* ha pretendido en todo momento aplicar los conceptos impartidos en la asignatura.

Por un lado, de cara a obtener los enlaces a las diferentes plataformas de videojuegos de consola, se ha construido una función modularizada 'link_crawler' empleando el material didáctico complementario de la asignatura (Penman, 2015), como puede observarse:

```
def link_crawler(seed_url, link_regex):
    crawl_queue = [seed_url]
    visited = set()
    enlaces = []

    while crawl_queue:
        url = crawl_queue.pop()
```

```
        if url in visited:
            continue
        visited.add(url)

        html = download(url)
        if html is None:
            continue

        soup = BeautifulSoup(html, 'html.parser')
        for link in soup.find_all('a', href=True):
            href = link['href']
            full_url = urljoin(url, href)
            if re.match(link_regex, href):
                enlaces.append(full_url)
                print("Añadiendo enlace:", full_url)

    return enlaces
```

En cuanto a la función `download` incluida en el *Crawler*, observamos que maneja errores y reintentos. Establece un retardo entre peticiones y la posibilidad de modificar el 'user_agent', como puede observarse:

```
def download(url, user_agent='Mozilla/5.0', retries=2):
    print('Downloading:', url)
    headers = {'User-agent': user_agent}
    req = urllib.request.Request(url, headers=headers)

    try:
        # 'Tiempo fuera' para evitar bloqueos prolongados
        html = urllib.request.urlopen(req, timeout=10).read()
    except urllib.error.HTTPError as e:
        print('HTTP Error:', e.code, e.reason)
        html = None
    except urllib.request.URLError as e:
        print('Download error:', e.reason)
        html = None
        if retries > 0 and hasattr(e, 'code') and 500 <= e.code < 600:
            # Reintento para errores 5xx
            return download(url, user_agent, retries - 1)
    # Retraso para evitar bloqueos
    time.sleep(1)
    return html
```

Con base a los enlaces obtenidos por nuestro *link_crawler*, otra función crea un archivo *.csv*:

```
def generar_csv(enlaces, nombre_archivo):
    # Obtenemos la fecha actual en formato YYYYMMDD
    fecha_actual = strftime("%Y%m%d")
    # Construimos el nombre del archivo con la fecha
    nombre_archivo_con_fecha = f"{nombre_archivo[:-4]}_{fecha_actual}.csv"

    # Construimos la ruta completa al archivo en la carpeta 'pricecharting'
```

```
    ruta_archivo = os.path.join('pricecharting', nombre_archivo_con_fecha)
    df = pd.DataFrame(enlaces)
    # Guardamos el archivo CSV
    df.to_csv(ruta_archivo, index=False)
    print(f"Se han guardado {len(enlaces)} enlaces en el archivo
'{nombre_archivo_con_fecha}'.\n")
```

Extraemos el enlace desde el listado resultante:

```
# URL base de la página
# Filtramos los enlaces que contengan la palabra "xbox-one"
base_url =
enlaces_price_charting[enlaces_price_charting['enlaces'].str.contains("xbox-one",
case=False)]
base_url = base_url['enlaces'].iloc[0] if not base_url.empty else None
```

El principal problema a abordar fue que Pricecharting.com utiliza la técnica de "*scroll infinito*" para cargar los productos de forma dinámica a medida que el usuario se desplaza por la página. Esto complicó la extracción de datos, ya que no estaban disponibles en el HTML inicial.
Descargamos y procesamos los precios con otras dos funciones modulares utilizando `request_html` en una sesión HTML:

```
from requests_html import HTMLSession

def download_pc(url, session=HTMLSession()):
    # Parámetros base peticiones
    params = {
        "sort": "",
        "when": "none",
        "release-date": "2024-10-31",
        "format": "json"
    }

    # Valor inicial del cursor
    cursor = 0

    # Lista para almacenar los datos de los productos
    products_data = []

    # Bucle para cargar páginas de productos (videojuegos)
    while True:
        # Actualizamos el valor del cursor
        cursor += 50
        params["cursor"] = cursor

        # Realizamos la petición desde la URL base
        # con los parámetros de la petición
        response = session.get(url, params=params)

        # Si la petición es exitosa
```

```
        if response.status_code == 200:
            try:
                # Convertimos la respuesta a JSON
                data = response.json()

                # Si no hay más productos, salimos del bucle
                if not data["products"]:
                    break

                # Iteramos sobre los productos y extraemos los datos
                for product in data["products"]:
                    product_data = extract_product_data(product)
                    products_data.append(product_data)

            except ValueError:
                print(f"Error al procesar la respuesta JSON: {response.content}")
                break

        else:
            print(f"Error en la petición: {response.status_code}")
            break

    return products_data

def extract_product_data(product):
    product_id = product["id"]
    title = product["productName"]
    loose_price = product["price1"]
    cib_price = product["price2"]
    new_price = product["price3"]
    return product_id, title, loose_price, cib_price, new_price
```

La solución por tanto consistió en analizar el comportamiento de la página e identificar las peticiones 'XHR' enviadas al servidor para cargar más productos al hacer scroll. Estas peticiones se simularon con la librería Requests-HTML, lo que permitió obtener todos los datos de manera eficiente, como ha podido observarse en el extracto de los diferentes retazos de código, implementando los parámetros propios de las peticiones en una variable 'params' y llamando con `session.get(url, params=params)`:

```
[...]
params = {
        "sort": "",
        "when": "none",
        "release-date": "2024-10-31",
        "format": "json"
    }
[...]
response = session.get(url, params=params)
[...]
```

Con ayuda de otra función procesamos el contenido:

```
def extract_product_data(product):
    product_id = product["id"]
    title = product["productName"]
    loose_price = product["price1"]
    cib_price = product["price2"]
    new_price = product["price3"]
    return product_id, title, loose_price, cib_price, new_price
```

Con otra función creamos el dataset que incluye la fecha del sistema:

```
def generar_csv_precios_v3(nombre_archivo, datos, nombres_columnas, plataforma):
    fecha_actual = strftime("%Y%m%d")
    nombre_archivo_con_fecha = f"
{nombre_archivo[:-4]}_{plataforma}_{fecha_actual}.csv"

    # Obtenemos la ruta del directorio actual
    directorio_actual = os.path.dirname(os.path.abspath(__file__))

    # Construimos la ruta al directorio 'dataset' un nivel por encima
    ruta_dataset = os.path.join(directorio_actual, '..', '..', 'dataset')

    # Creamos el directorio 'dataset' si no existe
    os.makedirs(ruta_dataset, exist_ok=True)

    # Construimos la ruta completa al archivo en la carpeta 'dataset'
    ruta_archivo = os.path.join(ruta_dataset, nombre_archivo_con_fecha)

    print(f"Directorio actual: {directorio_actual}\n")
    print(f"Ruta final del archivo CSV: {ruta_archivo}\n")

    with open(ruta_archivo, 'w', newline='', encoding='utf-8') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(nombres_columnas)  # Escribimos la cabecera
        writer.writerows(datos)  # Escribimos los datos
        print(f"Se han guardado los datos en el archivo
'{nombre_archivo_con_fecha}'.\n")
```

A continuación, el script agrega una columna cuyo contenido depende del sistema de consola del que hayamos descargado los datos y se salva tanto en formato *.csv* como en un dataframe de `pandas`. También utilizamos funciones como `os` para guardar los resultados de acuerdo a los lineamentos de directorios propuestos en la practica.

El último paso sería crear un dataset unificado desde datasets de `pandas`:

```
df = pd.concat([df_precios_xbox_one, df_precios_xbox_series_x], ignore_index=True)
df.to_csv(os.path.join(ruta_dataset, 'videojuegos_xbox.csv'), index=False)
```

Concluyendo en un dataset que incluiría diferentes precios de todos los videojuegos de consolas *Xbox one* y *Xbox series X* disponibles en www.pricecharting.com/es/

La estructura de la página de Pricecharting.com es susceptible de cambiar en el futuro, lo cual podría afectar el funcionamiento del script de scraping. Para poder mantener el script funcional, se documentó detalladamente el proceso de scraping y se identificaron los elementos clave necesarios para la extracción de datos. Esto permitirá una rápida adaptación del script en caso de cambios en la estructura del sitio.

Se implementaron mecanismos para limitar la frecuencia de las solicitudes, incluyendo retardos entre las peticiones y la rotación de user-agents para simular distintos navegadores y reducir la probabilidad de ser bloqueado, aunque no hayamos encontrado dificultades ni bloqueos en el proceso.

Finalmente, tuvimos que abordar diversos problemas ya que suelo trabajar sobre 'Dev Containers' del IDE VSCode, aspecto este último que complicaba sobremanera el trabajo de scraping al tener que incorporar los "WebDrivers" al *stack*.

Repositorio en GitHub:
https://github.com/Kamaranis/Web-Scraping-de-videojuegos-con-potencial-de-revalorizacion.git

10. Dataset

- Barrera Mora, A. (2024). Listado de videojuegos de XBOX susceptibles de revalorización [Data set]. Zenodo. https://doi.org/10.5281/zenodo.14043146

11. Video

https://drive.google.com/file/d/1PB7gAi-zFeUey35RGmEGJwlCjwKJDR8E/view?usp=drive_link

Bibliografía

- Barrera Mora, A. (2024). Listado de videojuegos de XBOX susceptibles de revalorización [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.14043146
- Chouinard, J.-C. (2023, junio 2). Web Scraping With Python and Requests-HTML (with Example). JC Chouinard. https://www.jcchouinard.com/web-scraping-with-python-and-requests-html/
- FAQ and Help for PriceCharting. (s. f.). PriceCharting. Recuperado 6 de noviembre de 2024, de https://www.pricecharting.com/faq#prices
- Goh, E., Al-Tabbaa, O., & Khan, Z. (2023). Unravelling the complexity of the Video Game Industry: An integrative framework and future research directions. Telematics and Informatics Reports, 12, 100100. https://doi.org/10.1016/j.teler.2023.100100
- Penman, R. (2015). Web scraping with python : Successfully scrape data from any website with the power of python. Packt Publishing, Limited.
- PriceCharting | Price Guides for Games, Cards & Comics. (s. f.). PriceCharting. Recuperado 6 de noviembre de 2024, de https://www.pricecharting.com/
- Requests-HTML: HTML Parsing for Humans (writing Python 3)! —Requests-HTML v0.3.4 documentation. (s. f.). Recuperado 6 de noviembre de 2024, de https://requests-html.kennethreitz.org/

- Writer, K. T. / Y. S. S. (2023, febrero 18). Japan Vintage Video Games Growing in Popularity Among Youngsters, Foreigners. https://japannews.yomiuri.co.jp/business/economy/20230218-92012/