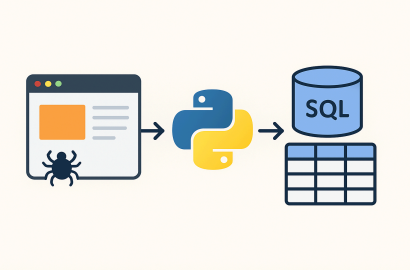
OBJETIVO

Para este proyecto final el estudiante deberá crear un web scraper que recopile información sobre un sitio de E-commerce que cuenta con diferentes productos.

Posteriormente guardaremos esa información en un repositorio alojado en Azure, lo que nos permitirá analizar los datos o visualizarlos en otras aplicaciones.



A continuación se detallan las etapas a realizar en la práctica:

1. Preparar el notebook para la extracción

Lo primero que debe realizarse es abrir un nuevo **notebook en Colab** e instalar las librerías necesarias para realizar la práctica. Algunas de las librerías que va a necesitar:

- **Requests**: para descargar el contenido de la página web.
- **BeautifulSoup** (bs4): para analizar el HTML y extraer información.
- **Pandas**: para organizar los datos en forma de tabla
- **Sqlalchemy** y **pymssql:** Para conectarnos con la base de datos y guardar la información que obtuvimos.

a. Explorar y cargar la página en el notebook.

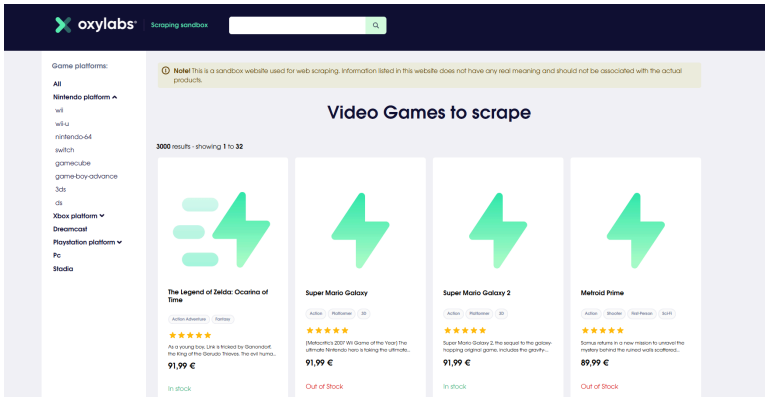
Actividad: Usando Google Colab, incluya en el notebook las celdas que le permiten extraer información de la página objetivo, validar la petición o request y probar que está extrayendo la información correctamente

La página a la cual se le va a hacer scraping es la siguiente:

E-commerce | Oxylabs Scraping Sandbox

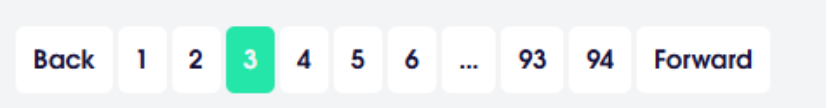
Esta página contiene un E-commerce creado específicamente para hacer Scraping. Para esta práctica, el objetivo es obtener la información de cada uno de los productos de la página web.

La página se ve así:



Tenga en cuenta que el catálogo contiene diferentes páginas de productos. Entonces, es necesario consultar las páginas a través de los links para obtener la información. En la parte inferior de la página se encuentran los enlaces para desplazarse entre las páginas del catálogo.

Para el proyecto final será necesario obtener la información por lo menos de 5 páginas diferentes.



Explore y entienda la estructura del sitio web usando las herramientas de developer o inspeccionando el código fuente(CTRL + Shift + I)

b. Entendiendo la estructura de la página.

Como en todo proyecto de scraping, es importante entender la página Web de la cual se va a obtener la información. Usando las herramientas de desarrollo o de inspección del navegador, explore la estructura y diseñe las estrategias para obtener los diferentes productos. De cada producto se quiere obtener:

- Título
- Géneros
- Descripción
- Precio
- Desarrollador
- Tipo
- URL (Esta es la URL utilizada para encontrar el producto en específico).

Para estos últimos 3 atributos, será necesario navegar dentro del producto para poder obtener la información. (Para poder hacer esto utilizaremos el href del elemento <a> que cada producto tiene)

NOTA: Vamos a excluir las estrellas de los datos recolectados porque se cargan dinámicamente con JavaScript, y esta tarea queda por fuera del alcance de la actividad.

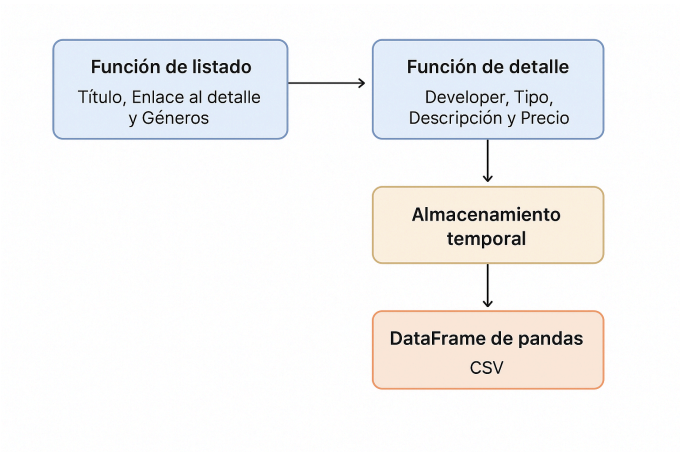
Actividad: Incluya en el notebook las celdas que le permiten consultar algunos de los elementos de la página objetivo

2. Estrategia de extracción

Considerando la estructura de la página objetivo y la información requerida para esta tarea, se propone la siguiente estrategia de extracción. Usted puede seguirla paso a paso o proponer e implementar una mejora.

El proceso completo de scraping y almacenamiento se compone de **dos funciones principales** y **dos etapas de almacenamiento**:

1. **Función de listado:**
   Recorre las páginas de productos y extrae información básica, como **título**, **enlace al detalle, descripción** y **géneros**.
   Los resultados se guardan temporalmente en una **lista de diccionarios**.
2. **Función de detalle:**
   A partir de los enlaces obtenidos, visita cada página individual para extraer información más específica, como **developer**, **tipo**, **descripción** y **precio**.
   Esta información se combina con la del listado para formar una estructura de datos completa.



3. **Almacenamiento temporal:**
   Los datos combinados se conservan en **diccionarios** dentro de una lista.
4. **Almacenamiento final:**
   Toda la información consolidada se transforma en un **DataFrame de pandas**, que luego puede **exportarse a un archivo CSV** para su análisis o carga en una base de datos.

Paso 1 : Función de listado

Esta función se encarga de recorrer las páginas de listados de productos y extraer la información general de cada uno. El objetivo es recorrer **las páginas de productos (listados)** y extraer la información básica de cada uno: **Título, Enlace al detalle y Géneros**

Pasos para hacerlo:

1. **Obtener el HTML** de una página de listado con requests.get(url).
   - Si la petición es exitosa, obtendrás un texto largo (el código HTML).
   - Usa `response.raise_for_status()` para detectar errores de conexión. Con esto se asegura de que si hay un error en la conexión (como una página no encontrada o un problema de red), el scraper lo detecte y lo maneje apropiadamente.

2. **Analizar el HTML con BeautifulSoup**.
   - Convierte el texto en un objeto BeautifulSoup para poder navegar por su estructura: soup = BeautifulSoup(response.text, "html.parser").
   - Este objeto soup representa el árbol del HTML y permite buscar etiquetas, atributos y contenidos de manera estructurada.

3. **Localizar los productos en la página**.
   - Cada producto está dentro de un bloque HTML como el siguiente:

```
<div class="product-card">
  ...
</div>
```

- Usa `.find_all("div", class_="product-card ...")`.
- Esto devuelve una lista de objetos o elementos `div` que representan cada producto en la página.

4. **Extraer la información de cada producto**.
   Recorre la lista de productos y extrae la información relevante para cada uno:
   - **Título:** se busca un <h4 class="title">. Si no existe, guardar "N/A".
   - **Enlace:** se extrae desde la etiqueta <a class="card-header">.
     - Si el enlace es relativo (ejemplo: /products/1), se debe convertir en absoluto (ejemplo: https://sandbox.oxylabs.io/products/1).
   - **Géneros:** se seleccionan todas las etiquetas dentro de <p class="category"> usando select().
     - Usa `p.select("p.category span")` para obtener todos.
     - Convierte la lista de géneros en un string con ", ".join(lista).

   Nota: El método `.select()` de BeautifulSoup es muy útil, permite seleccionar elementos utilizando un selector CSS, lo cual es muy útil para seleccionar múltiples elementos a la vez, como en este caso los géneros del producto.

5. **Almacenar la información en una lista de lista (lista de diccionarios)**.
   Cada producto se almacena como un diccionario como se ilustra a continuación:

```
{
  "titulo": "Ejemplo de Juego",
  "url": "https://sandbox.oxylabs.io/products/1",
  "generos": "Acción, Aventura"
}
```

Con esta función ya tenemos los datos básicos para cada producto, que nos servirán para visitar las páginas individuales en el siguiente paso.

Paso 2: Función de detalle

Con la información obtenida en el paso anterior, esta función —utilizando un **bucle `for`**— visita individualmente cada enlace de producto para recopilar los datos faltantes: **desarrollador**, **tipo**, **descripción** y **precio**.

Pasos para hacerlo:

1. **Obtener el HTML del detalle del producto** con requests.get(url) y convertirlo en BeautifulSoup.
2. **Extraer la información campo por campo**: Cada campo se encuentra dentro de una etiqueta específica. Algunos elementos pueden cambiar de posición o no estar presentes, por lo que es importante incluir validaciones y valores por defecto.
    - **Developer:** se encuentra dentro de un span.brand.developer. A veces aparece con el texto "Developer: Nombre", por eso debemos limpiar el texto eliminando "Developer:".
    - **Type:** está dentro de div.brand-wrapper span. No siempre está en la misma posición, así que conviene recorrer todos los de esa sección y buscar el que contiene "Type:".
    - **Descripción:** se encuentra dentro de un <p class="description">.
    - **Precio:** ubicado dentro de un .
    - **Estrellas (opcional):** a veces se representan con íconos <svg> dentro de div.star-rating. Contarlos nos da el número de estrellas.
3. **Manejo de datos faltantes**.
   Es posible que en algunos productos falte un campo (ejemplo: un producto gratis sin precio).
    - En esos casos usamos un valor por defecto como "N/A".
    - Esto hace que el scraper sea más **robusto** y tolerante a inconsistencias.
4. **Guardar la información en un diccionario**.
   Ejemplo:

```
{
  "Título": "Juego Ejemplo",
  "Developer": "Studio XYZ",
  "Type": "Game",
  "Descripción": "Juego épico de aventuras",
  "Precio": "$19.99",
  "URL": "https://sandbox.oxylabs.io/products/1"
}
```

La idea es que aquí vemos la diferencia entre un **scraping superficial** (sacar solo lo que hay en los listados) y un **scraping profundo** (entrar al detalle de cada producto).

Paso 3: Combinar la información

La idea es **combinar la información básica obtenida de los datos generales con los datos detallados** extraídos de cada producto.

El resultado se almacena en una **lista de diccionarios**, que posteriormente se transforma en un **DataFrame**.

De esta manera, las funciones desarrolladas anteriormente se integran en el **flujo principal del bot de scraping**, permitiendo construir la estructura final que contiene todos los productos con su información completa.

Para combinar las listas o diccionarios y preparar los datos antes de crear el DataFrame te sugerimos consultar:

- Python Official Docs – Data Structures
  Allí encontrarás explicaciones y ejemplos sobre cómo trabajar con **listas y diccionarios**, estructuras ideales para almacenar los resultados combinados de tu scraping.

Paso 4: Guardar en un DataFrame y exportar a CSV

Pandas DataFrame te permitirá analizar, filtrar o exportar la información fácilmente.

> **Ventaja:** Un DataFrame es una estructura de datos similar a una hoja de cálculo, ideal para análisis posteriores y exportación.

Esto nos da la ventaja de ver los primeros registros con df.head() y corroborar información

	Título	URL	Generos	Developer	Type	Descripción	Precio
0	The Legend of Zelda: Ocarina of Time	https://sandbox.oxylabs.io/products/1	Action Adventure, Fantasy	Nintendo	singleplayer	As a young boy, Link is tricked by Ganondorf, ...	91,99 €
1	Super Mario Galaxy	https://sandbox.oxylabs.io/products/2	Action, Platformer, 3D	Nintendo	singleplayer	[Metacritic's 2007 Wii Game of the Year] The u...	91,99 €
2	Super Mario Galaxy 2	https://sandbox.oxylabs.io/products/3	Action, Platformer, 3D	Nintendo EAD Tokyo	singleplayer	Super Mario Galaxy 2, the sequel to the galaxy...	91,99 €
3	Metroid Prime	https://sandbox.oxylabs.io/products/4	Action, Shooter, First-Person, Sci-Fi	Retro Studios	singleplayer	Samus returns in a new mission to unravel the ...	89,99 €
4	Super Mario Odyssey	https://sandbox.oxylabs.io/products/5	Action, Platformer, 3D	Nintendo	singleplayer	New Evolution of Mario Sandbox-Style Gameplay....	89,99 €

El dataframe se debería ver como el de la imagen.

Para convertir la lista de diccionarios en un DataFrame y guardar los resultados te sugerimos consultar:

- pandas.DataFrame — Documentación oficial
  Cómo crear un DataFrame a partir de una lista de diccionarios.
- pandas.DataFrame.to_csv — Exportar a CSV
  Explica cómo guardar tu DataFrame en un archivo CSV.
- 10 minutos con pandas (guía oficial)
  Ejemplos rápidos de creación, visualización y exportación de DataFrames.

Exportar los resultados a un archivo CSV: `productos.csv`.

3. Cargando el archivo en Azure Blob Storage

Enviar el archivo `productos.csv` generado en el paso anterior a un **contenedor de Azure Blob Storage** llamado `scraping-data`, donde podrá almacenarse de forma segura y accesible para otros sistemas o análisis posteriores.

Como vimos en la unidad anterior, un Azure **Blob Storage** es un servicio de almacenamiento en la nube que permite guardar archivos de todo tipo (CSV, imágenes, JSON, etc.) dentro de **contenedores** organizados.

Podría realizarse directamente desde el Google Colab, o realizando el proceso de ETL como se trabajó en la Unidad 2.

Para hacerlo desde Python

Desde Python, podemos interactuar con Azure Blob usando la librería `azure-storage-blob`. Instalando la librería de Azure:

```
!pip install azure-storage-blob
```

Consulta cómo puedes hacer para terminar esta tarea desde tu notebook en **Colab** y usando Python.

4. Verificar los resultados en Azure (Se realiza en el portal de Azure)

Después del paso anterior, los datos ya estarán almacenados en la nube. Entra al **portal de Azure** → **Storage Account** → **Containers** → **`scraping-data`**

y confirma que el archivo `productos.csv` aparece con la fecha y hora de carga.