THE BRIDGE

WebScrapping
Introduction

1. Introducción: ¿Qué es Web Scraping?

- **Definición Simple:** Técnica utilizada para extraer grandes cantidades de datos de sitios web de manera automatizada.
- **Metáfora:** Imagina un "robot" (el scraper) que lee las páginas web (como un humano) y copia la información que le interesa, guardándola en un formato.
- **Diferencia clave:** Transforma datos **no estructurados** (HTML) en datos **estructurados** (CSV, JSON, base de datos).

2. ¿Por Qué es Importante?

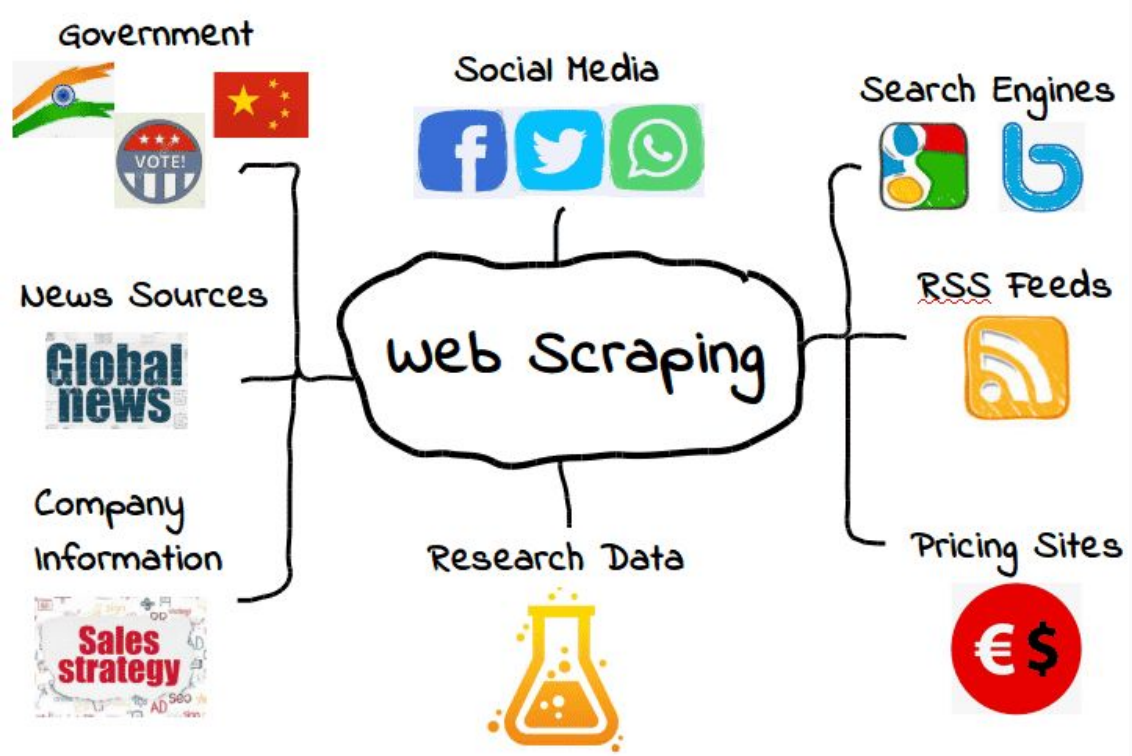
Comparación de Precios: Monitorear competidores en eCommerce.

Investigación de Mercados: Recopilar datos de productos, opiniones, o tendencias.

Generación de Leads: Extraer información de contacto de directorios públicos.

Noticias y Contenido: Seguimiento de artículos y *feeds* de información.

Análisis de Datos: Alimentar modelos de Machine Learning o *Business Intelligence*.



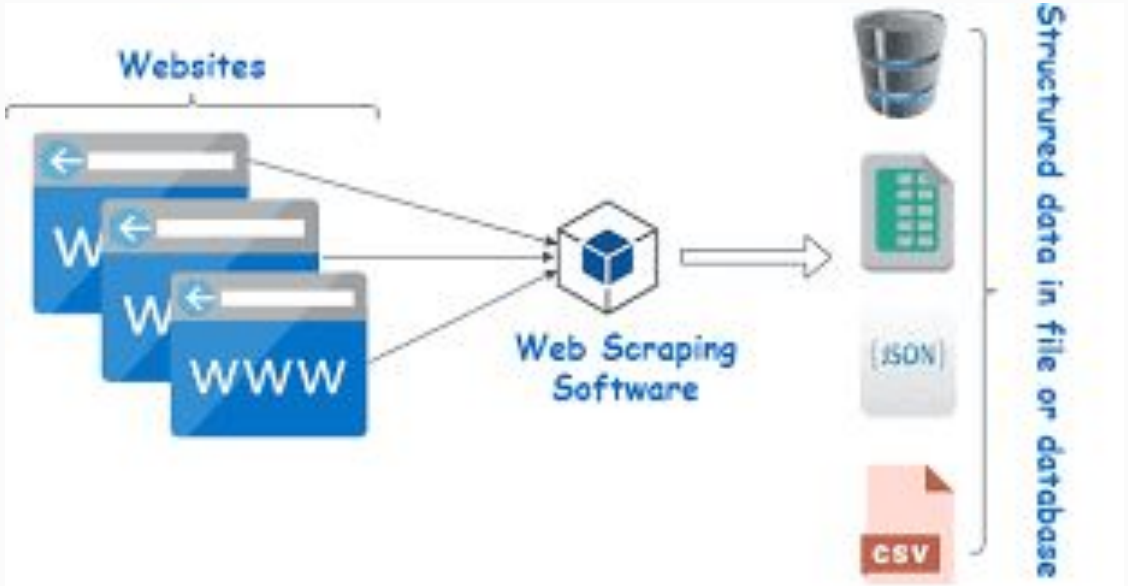
3. Fundamentos Técnicos (El Entorno Web)

HTML (Estructura): La columna vertebral de la página. El scraper busca etiquetas y atributos HTML.

CSS (Estilo): Aunque se usa para estilo, los **selectores CSS** son vitales para apuntar a los datos.

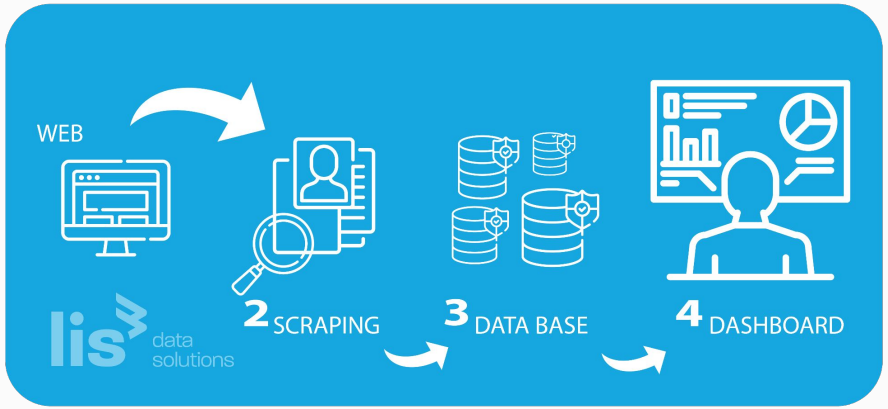
Peticiones HTTP: El scraper hace una solicitud (`GET` request) al servidor para obtener el código HTML de la página.

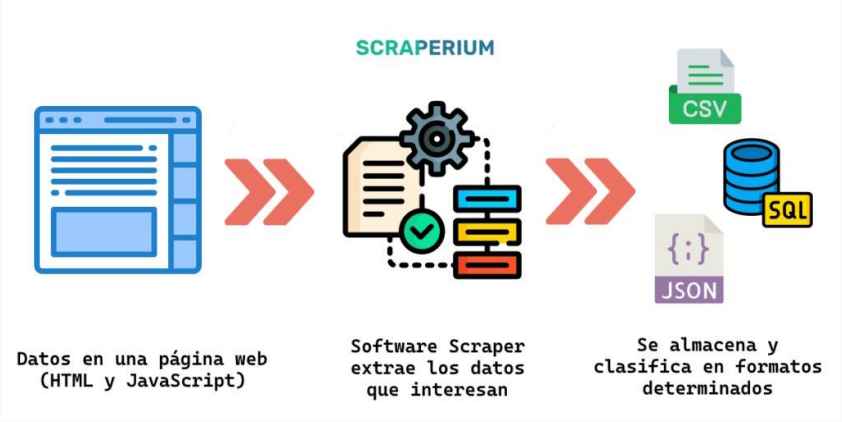
Robots.txt: Mencionar brevemente el archivo que indica a los bots qué áreas pueden o no rastrear.



4. El Proceso de Web Scraping (Paso a Paso)

1. **Planificación y Ética:** Elegir el objetivo, verificar `robots.txt` y Términos de Servicio.
2. **Análisis de la Web:** Inspeccionar el código HTML para identificar los datos deseados (usando las DevTools del navegador).
3. **Hacer la Petición:** Solicitar la URL para obtener el código fuente.
4. **Parseo/Análisis:** Usar librerías para navegar por el HTML y seleccionar los datos.
5. **Extracción de Datos:** Obtener el texto o valor de los elementos seleccionados.
6. **Almacenamiento:** Guardar los datos en CSV, JSON o una base de datos.

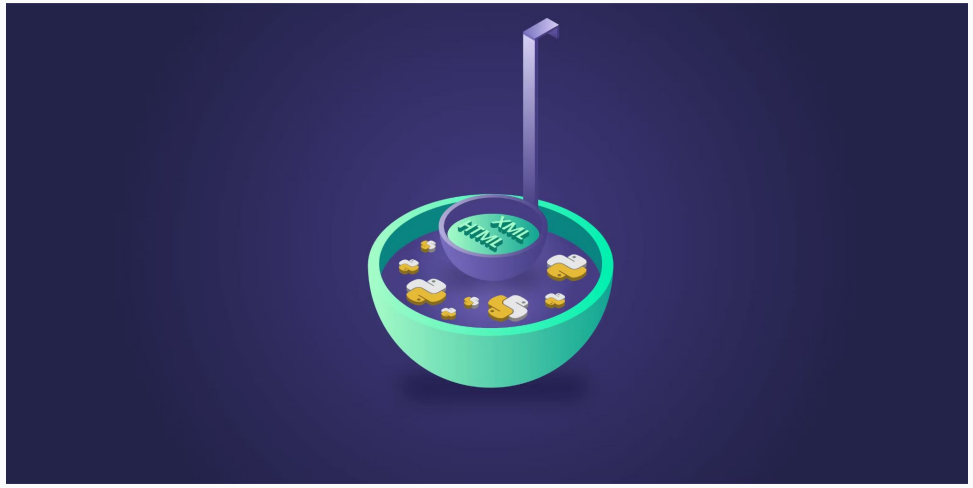


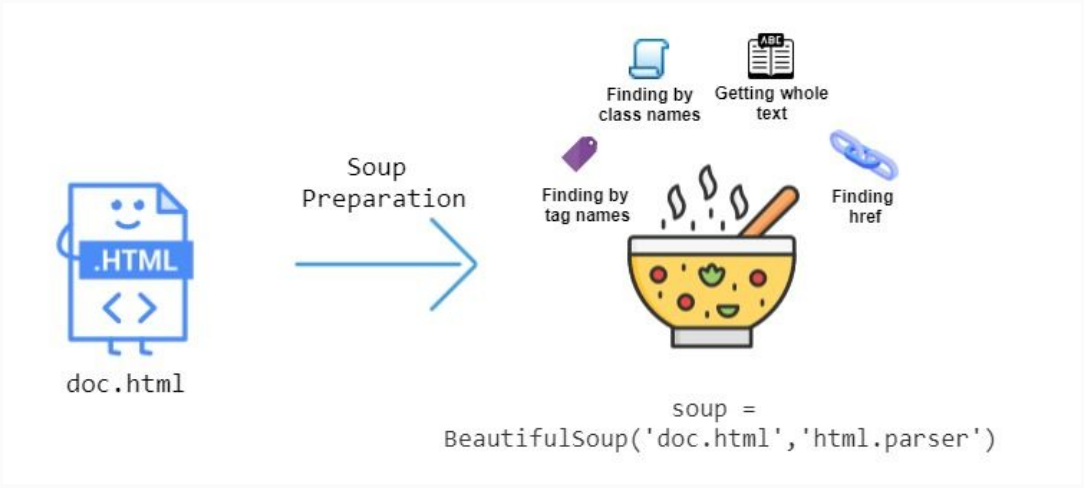


5. Herramientas Comunes

Python:

- **Requests:** Para hacer las peticiones HTTP.
- **BeautifulSoup:** Para el análisis (*parsing*) del HTML.
- **Scrapy:** Framework completo para proyectos grandes.



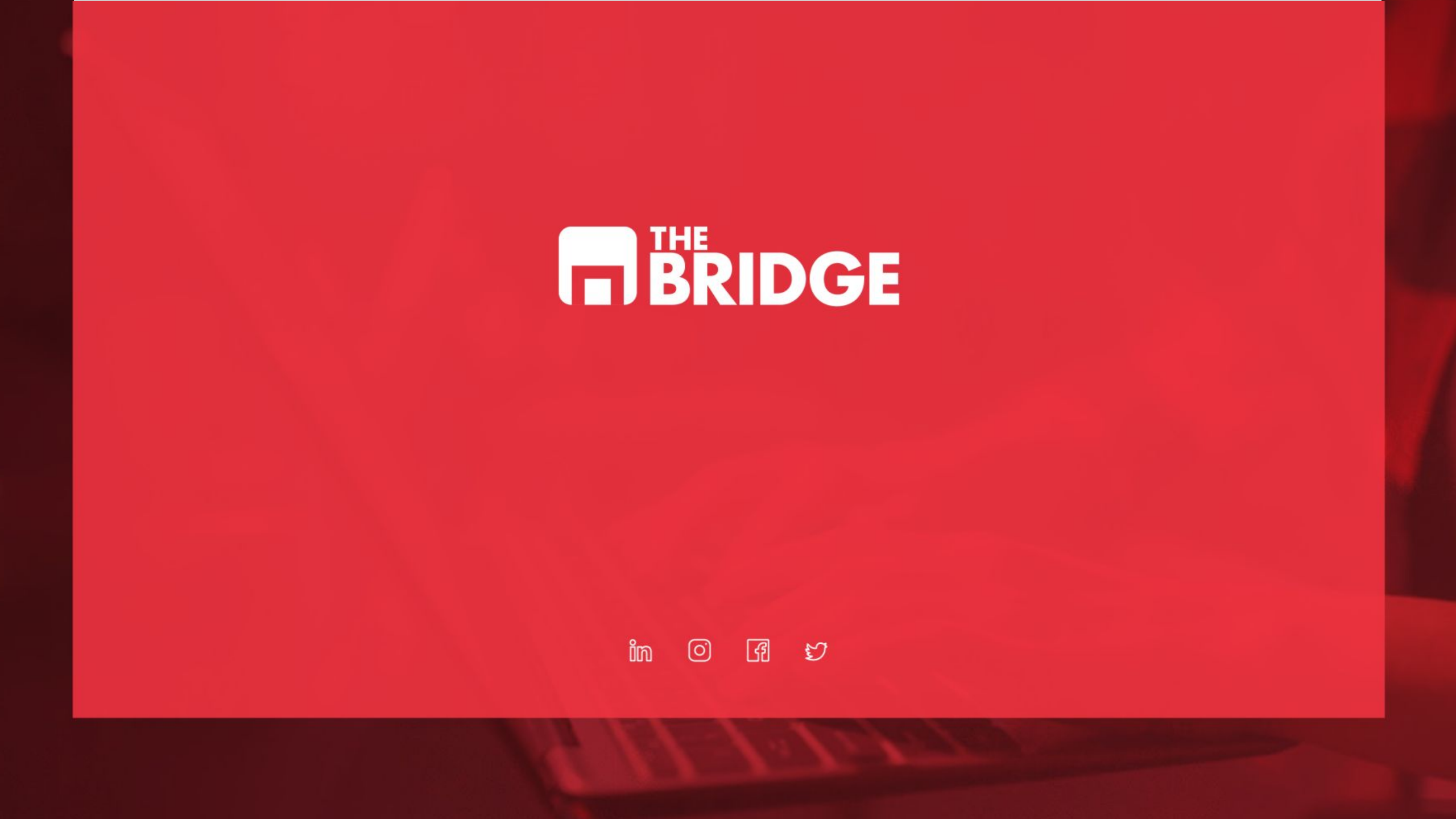


6. Ejemplo Práctico (Demo o Código Simple)

```python
import requests
from bs4 import BeautifulSoup

# 1. Definir la URL de la página web que queremos scrapear
url = 'http://quotes.toscrape.com/'

print(f"Scrapeando la URL: {url}\n")

try:
    # 2. Realizar una petición GET a la URL
    response = requests.get(url)
    response.raise_for_status()

    # 3. Parsear el contenido HTML con BeautifulSoup
    soup = BeautifulSoup(response.text, 'html.parser')

    # 4. Encontrar los elementos que contienen los datos que nos interesan
    quotes = soup.find_all('div', class_='quote')

    print("--- Citas Encontradas ---")
    # 5. Iterar sobre los elementos encontrados y extraer la información específica
    for i, quote in enumerate(quotes):
        text = quote.find('span', class_='text').text
        author = quote.find('small', class_='author').text

        tags_elements = quote.find('div', class_='tags').find_all('a', class_='tag')
        tags = [tag.text for tag in tags_elements]

        print(f"\nCita #{i+1}:")
        print(f"  Texto: {text}")
        print(f"  Autor: {author}")
        print(f"  Etiquetas: {', '.join(tags)}")

except requests.exceptions.RequestException as e:
    print(f"Error al conectar con la URL: {e}")
except Exception as e:
    print(f"Ocurrió un error: {e}")

print("\n--- Fin del Scraper ---")
```

7. Consideraciones Legales y Éticas

- **Revisar `robots.txt`:** Es la primera regla de cortesía. Si dice `Disallow`, respétalo.
- **Términos de Servicio (TdS):** Verificar si el sitio prohíbe explícitamente el scraping.
- **Privacidad de Datos:** Nunca extraer datos personales (GDPR, etc.) sin consentimiento explícito y base legal.
- **Carga del Servidor:** Implementar demoras (`time.sleep` o `Crawl-Delay`) entre peticiones para no saturar el sitio web.


THE
BRIDGE