# ÍNDICE

1. **Introducción.**
   1. **¿Qué es el web scrapping?**

El web scraping se refiere a la extracción de datos de un sitio web. Esta información se recopila y luego se exporta a un formato que sea más útil para el usuario. Ya sea una hoja de cálculo o una API.

Aunque el web scraping se puede hacer manualmente, en la mayoría de los casos, se prefieren las herramientas automatizadas cuando se extraen datos web, ya que pueden ser menos costosas y funcionan a un ritmo más rápido.

Diagrama

Descripción generada automáticamente

* 1. **¿Para qué sirve el web scrapping?**

Web Scraping tiene múltiples aplicaciones en diversas industrias. Veamos algunas de ellas, como por ejemplo, inteligencia de negocios, investigación de mercado, datos alternativos para las finanzas, bienes raíces, Monotorización de noticias y contenido, generación de leads, automatización empresarial, etc.

* 1. **¿Cómo funciona una web crapping?**

Los Web Scrapers pueden extraer todos los datos de sitios particulares o los datos específicos que desea un usuario. Idealmente, es mejor si especificas los datos que deseas para que el web scraper solo extraiga esos datos rápidamente. Por ejemplo, es posible que desees raspar una página de Amazon para los tipos de exprimidores disponibles, pero es posible que solo desees los datos sobre los modelos de diferentes exprimidores y no las opiniones de los clientes.

Entonces, cuando un web scraper necesita raspar un sitio, primero se le proporcionan las URL de los sitios requeridos. Luego carga todo el código HTML para esos sitios y un raspador más avanzado podría incluso extraer todos los elementos CSS y Javascript también.

Luego, el raspador obtiene los datos requeridos de este código HTML y genera estos datos en el formato especificado por el usuario. En su mayoría, se trata de una hoja de cálculo de Excel o un archivo CSV, pero los datos también se pueden guardar en otros formatos, como un archivo JSON.

1. **Implementación de código de Python para la extracción de datos de la página web de la sección amarilla.**
   1. **Agregamos las librerías a utilizar:**

En este caso, varias de estas librerías están enfocadas para poder realizar correctamente la ejecución de extracción de datos de una página web, como es el caso del la librería “BeautifulSoup”, Beautiful Soup es una biblioteca de Python para analizar documentos HTML. Esta biblioteca crea un árbol con todos los elementos del documento y puede ser utilizado para extraer información. Por lo tanto, esta biblioteca es útil para realizar web scraping — extraer información de sitios web.

Texto

Descripción generada automáticamente

* 1. **Agregamos un buscador del usuario y URLS a cargar:**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamentePara que diga qué es lo que quiere buscar en la página de la sección amarilla, en este caso, se usa un “user input”, que realiza este tipo de ejecución de buscador, además, hacemos la lectura de la URL de la página finalizando el URL con un for donde buscará en todas las secciones de la página, en este caso, como la página de la sección amarilla tiene solamente 30 secciones, entonces hace la búsqueda total de 1 a 31 implementado en el for.

* 1. **Usamos una función que extrae los campos de un LI**

Donde genera una fila de 8 campos, ya que se nos está pidiendo todos los datos de la página de la sección amarilla, donde nos pide el nombre, calle, municipio, ciudad, estado, código postal, número de teléfono y categoría, así como se ilustra en la siguiente imagen

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteEn este caso, para poder visualizar, qué codificación tiene cada uno de los siguientes parámetros, se debe de usar la herramienta conocida como “inspección de página”, que lo que hace es mostrar el código, en este caso, el HTML de cada una de las casillas a extraer la información, como se ilustra en la siguiente imagen.

Para después, construir el algoritmo de la función, para está pagina, el objetivo es extraer todos los datos de la sección mediante un concepto de búsqueda, en este caso, se usó buscar todas las ferreterías, y con la ayuda de la inspección de página, nos damos cuenta que el código o la línea de HTML que hace la búsqueda de cada uno de los elementos es la línea “listaDIV.find('span',itemprop='name')”, y la línea de html ‘span’, que agarra toda la información de la sección de la página.

Imagen que contiene Texto

Descripción generada automáticamentecambiando el ‘name’, para las demás creaciones a buscar, por ejemplo itemprop='address', itemprop='Region', etc, y para que todo se vaya leyendo, se implementa un if-else-if, donde va leyendo y extrayendo todas las casillas de búsqueda, y regresemamos todas las casillas con su información extraída con un return.

* 1. **Iteración de todas las urls (30 páginas).**

Para esto, se creó un arreglo vacío, donde con la función for, vamos a iterar los pasos que realizamos en la anterior sección, y vamos con “i”, guardando todo lo que tenemos con “i+=1”, después, Extraemos todo el contenido html para así, hacer el scrapping de los tags de html, Se busca el elemento ul con la función de la librería beautifulsoup “soupfind”, busca la lista de los li elementos encontrados totales, y hacemos un for donde creamos para la función “listaLi”, y crea un arreglo a base del anterior donde iteró todos los datos para agregarlo en las “6 columnas obtenidas” para finalizar se extrae la fila de datos del DIV listaDIV y se agrega a la lista FilasDataSet

Texto

Descripción generada automáticamente con confianza media

1. Creamos un dataframe donde almacene las filas vacías.
   1. En este caso, se crea con la función Dataframe, se usa header y se agregan todos los datos extraídos y creados.

header=['Nombre','Calle','Municipio','Ciudad','Estado','CodigoPostal','NumeroTelefono','Categoria']

**Y se imprime con la función pd.Dataframe y su función generadora.**

df\_ori=pd.DataFrame(FilasDataSet,columns=header)

Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteAdemás, si queremos ver cómo saldrá más adelante que hagamos la exportación, solamente usamos la función df\_ori (ya que este es el nombre que usamos para definir la creación).

* 1. ***Exportamos a un archivo .CSV (Excel) toda el scrapping hecho***.

Con la siguiente función,

**df\_ori.to\_csv(f'{user\_input}.csv',encoding='latin1',index=False)**

Donde to\_csv la hacemos llamar para hacer la salida a archivo de Excel, “user\_input” para que se llame el archivo como la búsqueda que hico el usuario, el encodign=’latin1’, para que palabras que tengan acento, pueda ponerlas y no generar ningún error, y el index=False, para que no enumere las columnas.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente