

26 DE ABRIL DE 2021

# REPORTE DE FUNCIONAMIENTO

## -PROYECTO-WEB SCRAPING CON FINES DE COMERCIO Y VENTAS-

- **Integrantes:**
  - Lorenzo Pérez Edna Yuritzzy
  - Rosas Garcia Eric Jonathan
- **Motivo:** Primera Estancia
- **Periodo:** Marzo-Abril 2021
- **Institución:** Universidad Politécnica Metropolitana de Hidalgo

## ***-Reporte Final de Funcionamiento-***

- **Objetivo:**

Diseñar y desarrollar una aplicación web para la recopilación de datos, a través de la técnica de web Scraping. Para esto se hará uso del lenguaje de programación Python, así como sus propias librerías y frameworks con los cuales se extraerán los datos de los sitios web proporcionados por la empresa. En específico se usará la librería de “Selenium” como motor principal del web scraping, apoyándose de bibliotecas de validación de entradas y organización de datos para realizar la tarea.

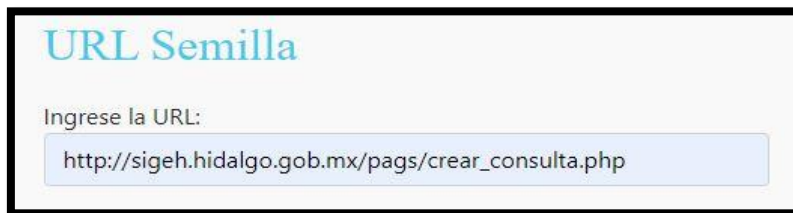
Se realizaron tres códigos, uno para cada uno de los tres sitios web proporcionados, aunque se utilizó la misma interfaz web para la extracción.

❖ Link : Pagina del SIGEH - Consulta de datos

- **Características:**

Cuenta con un sitio web donde puedes:

1. Introducir la URL de la página donde deseas hacer la extracción de datos.




The screenshot shows a web form titled "URL Semilla" in teal. Below the title, it says "Ingresa la URL:". There is a text input field containing the URL "http://sigeh.hidalgo.gob.mx/pags/crear\_consulta.php".

2. Elegir las opciones de qué datos deseas obtener de este sitio.



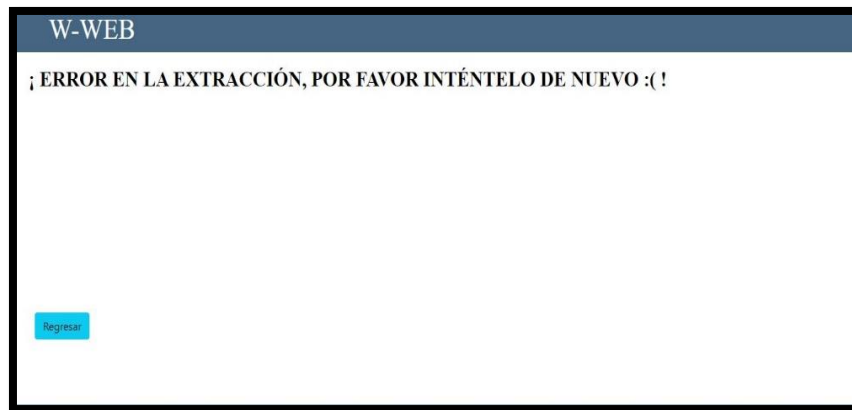
The screenshot shows a web form titled "Datos a extraer" in teal. It contains four dropdown menus for selecting data types: "Elige el primer tipo de dato a extraer:" (Títulos), "Elige el segundo tipo de dato a extraer:" (Fuentes), "Elige el tercer tipo de dato a extraer:" (Conceptos), and "Elige el cuarto tipo de dato a extraer:" (Valores). Below these is a text input field labeled "Ingresa el total de ítems a extraer :" with the number "3" entered.

3. Elegir el tipo de formato en que deseas obtener la información.



The screenshot shows a dropdown menu titled "Elige el formato del archivo a generar:". The menu is open, showing options: "-Abrir menú de selección-", CSV, JSON, TXT, Hoja de cálculo(XLSX), and Documento de Word(DOCX). The first option is highlighted in blue.

4. En caso de que por alguna razón el código no pueda procesar la información que el usuario está solicitando, el sitio web enviará “ERROR”.



5. Una vez que el sitio termina la extracción de datos, se genera en automático el archivo descargable.

El código de Python cuenta con las características:

1. La primera parte del código es la importación de librerías que estamos utilizando.

```
Codigo-WS.py
1  from selenium import webdriver
2  from selenium.webdriver.common.by import By
3  from selenium.webdriver.support import expected_conditions as EC
4  from selenium.webdriver.support.ui import WebDriverWait
5
6  import urllib
7  from urllib import request
8  from urllib.parse import urlparse
9  from flask import Flask, render_template, redirect, url_for, request
10 from selenium.webdriver.chrome.options import Options
```

Después están las rutas con las que el código ejecutará las páginas HTML como interfaz cuando se ejecute el código. Estas se ejecutan usando el framework de “Flask”.

```
app=Flask(__name__)
@app.route('/')
def home():
    return render_template('SitioWeb.html')

@app.route('/extraccion', methods=['POST', 'GET'])
def extraccion():
    liga=request.form['link']
    xpath1=request.form['xpath1']
    xpath2=request.form['xpath2']
    xpath3=request.form['xpath3']
    xpath4=request.form['xpath4']
    nItems=request.form['numberI']
    archivo=request.form['docdest']
```

- Debido a que desarrollamos web scraping y puede ser que al momento de pedirle al código que realice extracciones de datos, exista algún tipo de error colocamos desde el inicio un try, y un "except", la cual va a permitir que el código se ejecute, pero al momento de que exista algún error que nos imprima el error y no cause problemas con la extracción, principalmente que no nos baneen la dirección IP. Colocamos un contador que nos permita conocer en qué posición se encuentra el código extrayendo los datos, en este caso sería en qué municipio. Debido a que nItems es el número de municipios que se desea extraer, entonces si el usuario indica 3 el código realizara un mismo ciclo 3 veces. La variable selección contiene un valor string que suma el valor del contador más 2. Esto debido a que se usará más adelante para poder encontrar la expresión XPATH. La siguiente variable es un botón, el primer botón que nuestro código buscará, el cual cuenta con un único valor respecto a su expresión XPATH. Luego gracias a selenium podemos buscar donde se localiza y darle clic, seguido de eso, el código va a esperar un tiempo de 10 segundos para localizar todos los elementos que cumplan con una determinada expresión. cuando los encuentre seguirá con el código, y ahora da clic al botón 2, y en el archivo .csv va a escribir el texto que existe en ese botón, la cual es el nombre del municipio.

```
validadorNItems=nItems.isdigit()
partes=urlparse(liga)
if(partes.scheme==""):
    return render_template('ErrorExtraccion.html')
resp=urllib.request.urlopen(liga)
validacionU=resp.code
if(validacionU!=200):
    return render_template('ErrorExtraccion.html')
if(validadorNItems==False):
    return render_template('ErrorExtraccion.html')
nItems=int(nItems)
chrome_options=Options()
chrome_options.add_argument("--headless")
driver = webdriver.Chrome(chrome_options=chrome_options,executable_path=r"./chromedriver.exe")
driver.get(liga)
contador =0
documento="ArchivoE"+archivo
DE= open(documento,"w")
```

```
Codigo-WS.py
15
16     try:
17
18         while contador < nItems:
19
20
21             seleccion = str(contador+2)
22             boton = driver.find_element_by_xpath('//div[@class="container landing-wrapper"]/div/ul/li[1]')
23             boton.click()
24             WebDriverWait(driver,10).until(
25                 EC.presence_of_all_elements_located((By.XPATH, '//div[@class="container landing-wrapper"]'))
26             ))
27
28
29             boton2 = driver.find_element_by_xpath('//div[@class="container landing-wrapper"]/div/div/div/form/div/div/sel
30             boton2.click()
31             DE.write(boton2.text)
32             DE.write("\n")
33
34             botones= driver.find_elements_by_xpath('//div[@class="container landing-wrapper"]/div/div/div/form/div/div/di
35             for i in range(len(botones)):
36                 boton3 = botones[i]
```

Después de eso se va a dar un tiempo para poder encontrar el botón que de crear consulta, luego de dar clic en este, va a tomarse un tiempo en encontrar algunos elementos, puesto a que la página ha cambiado, y ahora realizará la extracción de datos. Encuentra todos los títulos y los almacena en una lista, hace lo mismo con las fuentes.

Después de encontrar todos los títulos y fuentes, se abrirá un ciclo for, para poder recorrer las 13 tablas que se generan desde la página “Crear consulta”, y de cada una de las tablas va a realizar las búsquedas, de los datos que el usuario desea extraer, si el usuario indicó que desea extraer los títulos, entonces el archivo csv va a escribir este dato, si pidió fuentes, estas se van a guardar.

```

Codigo-WS.py
9
10
11 titulos= driver.find_elements_by_xpath('//div[@class="container landing-wrapper"]/h2')
12
13 fuentes = driver.find_elements_by_xpath('//div[@class="container landing-wrapper"]/p')
14
15
16 for i in range(len(titulos)):
17     a= str(i+3)
18     if xpath1=="titulos" or xpath2=="titulos" or xpath3=="titulos" or xpath4=="titulos":
19         titulo= titulos[i].text
20         titulo = titulo.replace(',',' ')
21         DE.write(titulo)
22         DE.write("\n")
23     if xpath1=="fuentes" or xpath2=="fuentes" or xpath3=="fuentes" or xpath4=="fuentes":
24         fuente=fuentes[i].text
25         fuente= fuente.replace(',',' ')
26         DE.write(fuente)
27         DE.write("\n")
28     if xpath1=="conceptos" or xpath2=="conceptos" or xpath3=="conceptos" or xpath4=="conceptos":
29         if xpath1=="valores" or xpath2=="valores" or xpath3=="valores" or xpath4=="valores":
30             titolor= "Concepto,valor\n"
31             DE.write(titolor)
32
33
34 conceptos = driver.find_elements_by_xpath('//div[@class="container landing-wrapper"]/div['+str(a)+']+/')

```



Luego, almacenará todos los conceptos en una lista. se abrirá un ciclo, para que nos pueda generar una columna en cada tabla con respecto al número de conceptos que se tienen, debido a que conceptos y valores tienen el mismo número de datos, solo usamos a conceptos. Y realizamos la búsqueda de datos que el usuario desea extraer. Después de la extracción, guardará los resultados en el documento.

```
Codigo-WS.py
0  conceptos = driver.find_elements_by_xpath('//div[@class="container landing-wrapper"]/div+'+'+a+'+'+t
1
2
3      for j in range(len(conceptos)):
4          b = str(j+1)
5          if xpath1=="conceptos" or xpath2=="conceptos" or xpath3=="conceptos" or xpath4=="conceptos":
6              c = '//div[@class="container landing-wrapper"]/div+'+'+a+'+'+t+'/table/tbody/tr+'+'+b+'+'+t'/th'
7              concepto = driver.find_element_by_xpath(c).text
8              concepto = concepto.replace(',',' ')
9              if xpath1=="valores" or xpath2=="valores" or xpath3=="valores" or xpath4=="valores":
10                 c = '//div[@class="container landing-wrapper"]/div+'+'+a+'+'+t+'/table/tbody/tr+'+'+b+'+'+t'+'+'+t'
11                 concepto = driver.find_element_by_xpath(c).text
12                 concepto = concepto.replace(',',' ')
13                 d= '//div[@class="container landing-wrapper"]/div+'+'+a+'+'+t+'/table/tbody/tr+'+'+b+'+'+t'+'+'+t'
14                 valor = driver.find_element_by_xpath(d).text
15                 valor = valor.replace(',',' ')
16                 filas = concepto+" "+valor+"\n"
17                 DE.write(filas)
18             else:
19                 DE.write(concepto)
20                 DE.write("\n")
21         else:
```

Cuando termina de recorrer todas las tablas, el código procede a salir del ciclo y continuar, entonces busca al botón denominado “botón5”, le da clic y regresa nuevamente al sitio principal para volver a hacer el recorrido. De esta manera hasta que el contador suma la cantidad de municipios (ítems) que el usuario declaró que quería extraer.

```
Codigo-WS.py
29
30     boton5= driver.find_element_by_xpath('//div[@class="container landing-wrapper"]/div/article/h3/a')
31     boton5.click()
32
33     WebDriverWait(driver,8).until(
34         EC.presence_of_all_elements_located((By.XPATH, '//div[@class="container landing-wrapper"]')
35     ))
36
37     contador+=1
38     DE.write("\n")
39 except Exception as e:
40     print(e)
41     print('Error de extraccion')
42     return render_template('ErrorExtraccion.html')
43 driver.close()
44 print('Fin de la extraccion')
```

Para la parte final, solo se muestra el except que nos imprime el error y nos va a redirigir a la página de ERROR. Luego de eso, solo imprimirá “fin de la Extracción” y el código nos enviará a la página que nos muestra que la extracción ha finalizado.

```

Codigo-WS.py
8      DE.write("\n")
9  except Exception as e:
10     print(e)
11     print('Error de extraccion')
12     return render_template('ErrorExtraccion.html')
13 driver.close()
14 print('Fin de la extraccion')
15 return render_template('Extraccion.html', doc=documento)
16
17
18 if __name__ == '__main__':
19     app.run(debug=True)
20

```

❖ Link : Pagina del SIGEH - Indicadores

- **Características:**

Cuenta con el sitio web donde puede:

1. Ingresar la URL del sitio al que quieres acceder para extraer los datos.

## URL Semilla

Ingrese la URL:

2. Un apartado donde puedes elegir los datos a extraer.

## Datos a extraer

Elige el primer tipo de dato a extraer:

Títulos

Elige el segundo tipo de dato a extraer:

Fuentes

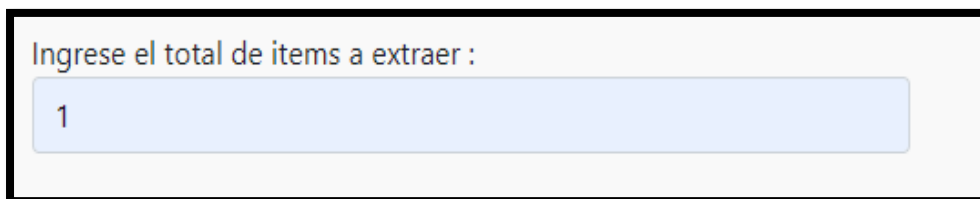
Elige el tercer tipo de dato a extraer:

Descripción

Elige el cuarto tipo de dato a extraer:

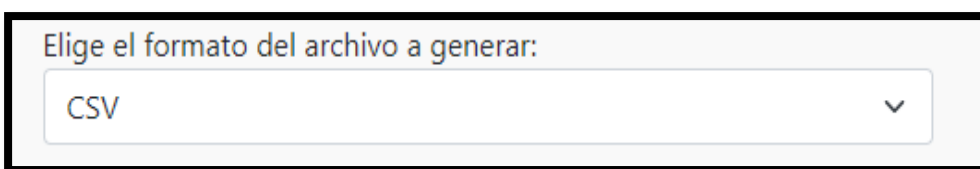
Temporalidad

3. La opción de elegir el número de indicadores o ítems a extraer.



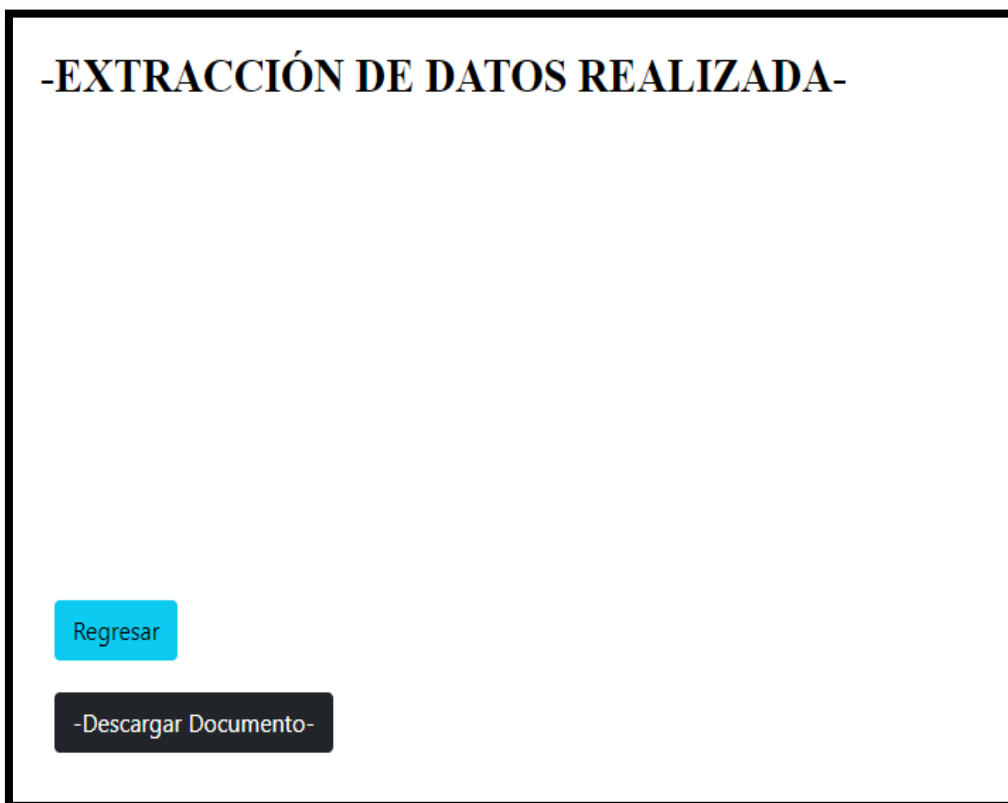
Form for selecting the number of items to extract. The label is "Ingrese el total de ítems a extraer :". The input field contains the number "1".

4. La opción de elegir el formato de salida del archivo donde se guardaran los datos extraídos.



Form for selecting the output file format. The label is "Elige el formato del archivo a generar:". The dropdown menu shows "CSV" and a downward arrow.

5. Una vez que se extraigan los datos, el sitio te lo indicará y permitirá repetir el proceso desde la página principal. En caso de que hubiera algún error, lo advertirá la página y pedirá volver a ingresar los datos.



Confirmation screen for data extraction. The main heading is "-EXTRACCIÓN DE DATOS REALIZADA-". At the bottom, there are two buttons: "Regresar" (Return) and "-Descargar Documento-" (Download Document).



El código de Python cuenta con las características:

1. La primera parte donde se importan todas las librerías a ocupar.

```
1  from time import sleep
2  from selenium import webdriver
3  import random
4  from selenium.webdriver.common.by import By
5  from selenium.webdriver.support import expected_conditions as EC
6  from selenium.webdriver.support.ui import WebDriverWait
7  from selenium.webdriver.chrome.options import Options
8  import urllib
9  from urllib import request
10 from urllib.parse import urlparse
11 from flask import Flask, render_template, redirect, url_for, request
12
```

2. Después se generan las rutas donde se retornan los HTML del sitio web. En la ruta de extracción de datos llamamos por medio del método “request” las entradas de los formularios de la página web, las cuales serán introducidas por el usuario.

```
app=Flask(__name__)
@app.route('/')
def home():
    return render_template('SitioWeb-Sigeh-Principal.html')

@app.route('/extraccion', methods=['POST', 'GET'])
def extraccion():
    liga=request.form['link']
    xpath1=request.form['xpath1']
    xpath2=request.form['xpath2']
    xpath3=request.form['xpath3']
    xpath4=request.form['xpath4']
    nItems=request.form['numberI']
    archivo=request.form['docdest']
```

3. Continuamos con la validación de la URL introducida y del número de ítems que se desean extraer, para esto usamos la librería “urllib” y el método “.isdigit()”.

```
validadorNItems=nItems.isdigit()
partes=urlparse(liga)
if(partes.scheme==""):
    return render_template('ErrorExtraccion.html')
resp=urllib.request.urlopen(liga)
validacionU=resp.code
if(validacionU!=200):
    return render_template('ErrorExtraccion.html')
if(validadorNItems==False):
    return render_template('ErrorExtraccion.html')
nItems=int(nItems)
```

4. Una vez que se validan las entradas, se genera el objeto “chrome\_options()” del tipo “Options()” el cual se encargará de ejecutar el navegador automatizado por selenium en segundo plano, evitando así interferencias manuales por parte del usuario, se le da la dirección del controlador en .exe y se le pasa por medio del método “.get()”. Después se abre un documento con la extensión que el usuario introdujo, y se le da la propiedad de write “w” con la cual reescribirá todo de nuevo siempre que se ejecute el código. Luego se entra al “try” donde se generará la búsqueda y extracción de los datos.

```
chrome_options = Options()
chrome_options.add_argument("--headless")
driver = webdriver.Chrome(chrome_options=chrome_options,executable_path=r"./chromedriver.exe")
driver.get(liga)
sleep(random.uniform(2.0, 3.0))
driver.refresh()
sleep(random.uniform(2.0, 3.0))
datos="DatosE-Principal-Sigeh"+archivo
documento = "docs/DatosE-Principal-Sigeh"+archivo
DE= open(documento,"w")
try:
    for i in range(nItems):
        sleep(random.uniform(3.0, 4.0))
        indicadores= driver.find_elements_by_xpath('//div[@class="col-lg-12"]/div/div/div/h4/a')
        a=str(i+2)
        ejes= driver.find_elements_by_xpath('//div[@class="container landing-wrapper"]/div['+a+']/div/div/div
```

5. El código iniciara desde una posición 0, en un rango menos 1 al dato que sea introducido por el usuario, lo primero que realizará es buscar a los indicadores/ejes y guardarlos en una lista, lo mismo para las opciones que despliega cada eje, luego de encontrarlos guardará el título, este título va a depender de la lista indicadores, luego guardará los datos en el archivo csv.

```
titulor= indicadores[i].text + ","+"Indicador, Fuente, Confiabilidad de la fuente, Descripcion, Tempor
DE.write(titulor)
for j in range(len(ejes)):
    indicadores= driver.find_elements_by_xpath('//div[@class="col-lg-12"]/div/div/div/h4/a')
    indicador= indicadores[i]
    sleep(random.uniform(3.0, 4.0))
    k= str(j+1)
    indicador.click()

    sleep(random.uniform(3.0, 5.0))
    indicador_consulta= driver.find_element_by_xpath('//div[@class="container landing-wrapper"]/div[
    indicador_consulta.click()
    sleep(random.uniform(2.0, 3.0))
    if xpath1=="titulos" or xpath2=="titulos" or xpath3=="titulos" or xpath4=="titulos":
        nombre_indicador= driver.find_element_by_xpath('//div[@class="container-fluid"]/div/div/div/
        fila1=" "+","+nombre_indicador.text
        DE.write(fila1)
```

- Después de guardar los datos, entrara a un for que se encargará de recorrer dentro de los ejes, en cada una de sus listas de datos. Pero, primero le damos clic al indicador para que pueda generar los nuevos links. Procedimos a recorrer dentro de cada uno de estos indicadores. Estando dentro de cada indicador, extraemos los datos que contienen, colocando algunos if's para encontrar los datos que el usuario desea extraer. Cada vez que el ciclo termina, busca el botón para regresar a la página principal y buscar nuevamente los ejes.

```
if xpath1=="fuentes" or xpath2=="fuentes" or xpath3=="fuentes" or xpath4=="fuentes":
    fuente_indicador= driver.find_element_by_xpath('//div[@class="container-fluid"]/div/div/div[6]/div/div/div[6])
    confiabilidad_de_la_fuente= driver.find_element_by_xpath('//div[@class="container-fluid"]/div/div/div[6]/div/div/div[7])
    fila2= ", "+fuente_indicador.text+", "+confiabilidad_de_la_fuente.text
    DE.write(fila2)

if xpath1=="conceptos" or xpath2=="conceptos" or xpath3=="conceptos" or xpath4=="conceptos":
    descripcion_indicador= driver.find_element_by_xpath('//div[@class="container-fluid"]/div/div/div[6]/div/div/div[8])
    fila3= ", "+descripcion_indicador.text
    DE.write(fila3)

if xpath1=="" or xpath2=="" or xpath3=="" or xpath4=="":
    temporalidad_indicador= driver.find_element_by_xpath('//div[@class="container-fluid"]/div/div/div[6]/div/div/div[9])
    fila4= ", "+temporalidad_indicador.text
    DE.write(fila4)

if xpath1=="valores" or xpath2=="valores" or xpath3=="valores" or xpath4=="valores":
    ultimo_dato= driver.find_element_by_xpath('//div[@class="container-fluid"]/div/div/div[6]/div/div/div[10])
    valor_hidraulico= driver.find_element_by_xpath('//div[@class="container-fluid"]/div/div/div[6]/div/div/div[11])
```

7. Terminamos con la parte del “except” el cual detendrá el código una vez que se encuentre un error y lo imprimirá en la línea de comandos así como retornara una página HTML advirtiéndole que hubo un fallo y que se intente de nuevo. En caso de que todo funcione correctamente, se detendrán/cerrarán el controlador del navegador y también el documento donde se guardaron los datos, para después retornar una página HTML indicando que todo corrió adecuadamente y que se puede descargar el archivo previamente generado con la data.
- Por último el “if \_\_name\_\_” sirve para mantener al sitio web actualizado mientras se modifica el código de python en las pruebas.

```
if xpath1=="valores" or xpath2=="valores" or xpath3=="valores" or xpath4=="valores":  
    ultimo_dato= driver.find_element_by_xpath('//div[@class="container-fluid"]/div/div/div[6]/div/div/div[6]  
    valor_hidalgo= driver.find_element_by_xpath('///div[@class="container-fluid"]/div/div/div[6]/div/div/div[6]  
    valor_nacional= driver.find_element_by_xpath('//div[@class="container-fluid"]/div/div/div[6]/div/div/div[6]  
    lugar_nacional= driver.find_element_by_xpath('///div[@class="container-fluid"]/div/div/div[6]/div/div/div[6]  
    filas = ","+ ultimo_dato.text+", "+valor_hidalgo.text+", "+valor_nacional.text+", "+lugar_nacional.text+"  
    DE.write(filas)  
  
boton5= driver.find_element_by_xpath('//div[@class="container landing-wrapper"]/div/article/h3/a')  
boton5.click()  
  
sleep(random.uniform(2.0, 4.0))
```

```

except Exception as e:
    print(e)
    print('Error de extraccion')
    return render_template('ErrorExtraccion.html')
driver.close()
print('Fin de la extraccion')
DE.close()
return render_template('Extraccion.html', doc=datos, dat=documento)

if __name__ == '__main__':
    app.run(debug=True)


```

❖ Link : Pagina del gobierno del estado de Hidalgo - Población

- **Características:**

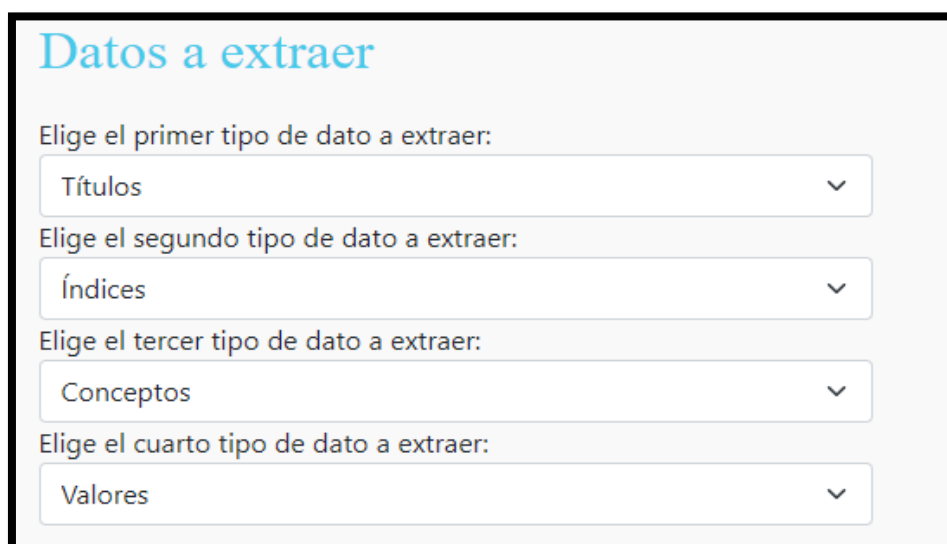
Cuenta con el sitio web donde puede:

1. Ingresar la URL del sitio de donde se extraerán los datos.



The screenshot shows a web form titled "URL Semilla" in blue text. Below the title, there is a label "Ingresa la URL:" followed by a text input field. The input field contains the URL "http://poblacion.hidalgo.gob.mx/".

2. Elegir el tipo de datos a extraer del sitio web.



The screenshot shows a web form titled "Datos a extraer" in blue text. Below the title, there are four labels, each followed by a dropdown menu:

- Elige el primer tipo de dato a extraer: (Dropdown menu showing "Títulos")
- Elige el segundo tipo de dato a extraer: (Dropdown menu showing "Índices")
- Elige el tercer tipo de dato a extraer: (Dropdown menu showing "Conceptos")
- Elige el cuarto tipo de dato a extraer: (Dropdown menu showing "Valores")

3. Ingresar la cantidad de ítems a extraer.

Ingrese el total de ítems a extraer :

4. Escoger el formato del archivo de salida que contendrá la data extraída.

Elige el formato del archivo a generar:

CSV ▼

5. Una vez que se introduzcan todos los datos y se de en “Extraer datos” el código se ejecutará, y mostrará alguna de dos páginas HTML, una en la que ocurrió un error durante la extracción y pide volverá ingresar los datos, y otra en la que se completó el proceso correctamente y permitirá al usuario descargar el archivo generado y regresar para volver a usar el sitio web.

## **-EXTRACCIÓN DE DATOS REALIZADA-**

Regresar

-Descargar Documento-

El código de Python cuenta con las siguientes características:

1. La primera parte donde se importan todas las librerías a utilizar.

```
1  from selenium import webdriver
2  from selenium.webdriver.common.by import By
3  from selenium.webdriver.support import expected_conditions as EC
4  from selenium.webdriver.support.ui import WebDriverWait
5  from selenium.webdriver.chrome.options import Options
6  from time import sleep
7  import random
8  import urllib
9  from urllib import request
10 from urllib.parse import urlparse
11 from flask import Flask, render_template, redirect, url_for, request
12
```

2. Después se encuentran las rutas que permiten ejecutar el sitio web a partir del código de python, retornando una página HTML cada una usando funciones. En la ruta de extracción estamos recibiendo las entradas que ingresa el usuario en los formularios del sitio, mediante el método “request.form[]”, para poder usarlos más adelante en el código de extracción.

```
13 app=Flask(__name__)
14 @app.route('/')
15 def home():
16     return render_template('SitioWeb-PoblacionHidalgo.html')
17
18 @app.route('/extraccion', methods=['POST', 'GET'])
19 def extraccion():
20     liga=request.form['link']
21     xpath1=request.form['xpath1']
22     xpath2=request.form['xpath2']
23     xpath3=request.form['xpath3']
24     xpath4=request.form['xpath4']
25     nItems=request.form['numberI']
26     archivo=request.form['docdest']
27
```



- Continuamos con la validación de la URL y el número de ítems a extraer ingresados por el usuario, esto apoyándonos de métodos de la librería “urllib” y del método “isdigit()”.

```
validadorNItems=nItems.isdigit()
partes=urlparse(liga)
if(partes.scheme==""):
    return render_template('ErrorExtraccion.html')
resp=urllib.request.urlopen(liga)
validacionU=resp.code
if(validacionU!=200):
    return render_template('ErrorExtraccion.html')
if(validadorNItems==False):
    return render_template('ErrorExtraccion.html')
nItems=int(nItems)
```

- Procedemos generando un objeto “chrome\_options” del tipo “Options()” el cual se encargará de ejecutar el navegador automatizado por selenium en segundo plano y así evitar interferencias manuales que pueda ocasionar el usuario. Generamos un objeto del tipo “webdriver.Chrome” el cual controlara el browser para acceder al sitio, y le pasamos la dirección del .exe en nuestro repositorio. Después generamos un documento con la extensión que nos pasara el usuario por medio del sitio, y lo generamos en modo write “w” lo cual reescribirá el documento cada vez que se ejecute el código.

```
chrome_options = Options()
chrome_options.add_argument("--headless")
driver = webdriver.Chrome(chrome_options=chrome_options, executable_path=r"./chromedriver.exe")
driver.get(liga)
datos="Informacion-Poblacion"+archivo
documento = "docs/Informacion-Poblacion"+archivo
DE= open(documento,"w")
try:
```

- Al iniciar con la extracción de los datos lo primero que realiza el código es, buscar el botón que permitirá entrar a otra página para obtener más información. Tomará un receso para encontrar otros elementos más tarde. Guardará en dos listas diferentes los datos de cantidades, que son los datos numéricos de los indicadores, y en otra variable se almacenarán los datos de los textos, que especifican los datos. Seguido de eso empezará a almacenar los índices, dentro de un IF en caso de que sea uno de los datos que el usuario esté requiriendo.

```

try:
    boton = WebDriverWait(driver,10).until(
        EC.presence_of_element_located((By.XPATH, '//*[@id="Modalveda"]/div/div/div[1]/button/span'))
    )
    boton.click()
    WebDriverWait(driver,10).until(
        EC.presence_of_all_elements_located((By.XPATH, '//div[@class="col-md-4 card-indicador"]/a'))
    ))
    cantidades=driver.find_elements_by_xpath('//div[@class="col-md-4 card-indicador"]/a/h2')
    parrafos=driver.find_elements_by_xpath('//div[@class="col-md-4 card-indicador"]/a/p')

    if xpath1=='indices' or xpath2=='indices' or xpath3=='indices' or xpath4=='indices':
        DE.write('Indices:'+'\n')
        for j in range(len(cantidades)):
            WebDriverWait(driver,20).until(
                EC.presence_of_element_located((By.XPATH, '//div[@class="col-md-4 card-indicador"]/a'))
            ))

```

6. Luego de eso, entra a un ciclo for, donde va a recorrer la lista de las cantidades, para obtener los datos de cada uno de ellos para posteriormente almacenarlos en el archivo csv. Tomará un tiempo para encontrar algunos datos, y después buscará el botón para eliminar la ventana emergente que aparece al entrar en cada una de las páginas de este sitio.

```

    cantidad = cantidades[j].text
    cantidad= cantidad.replace(',',' ')
    parrafo=parrafos[j].text
    parrafo=parrafo.replace(',',' ')
    DE.write(cantidad+', '+parrafo+'\n')

WebDriverWait(driver,10).until(
    EC.presence_of_all_elements_located((By.XPATH, '//div[@class="row row-gob row-body"]/div/div/article'))
)

boton1 = driver.find_element_by_xpath('//div[@class="row row-gob row-body"]/div/div/article[1]/h3/a')
boton1.click()

WebDriverWait(driver,6).until(
    EC.presence_of_all_elements_located((By.XPATH, '/html/body/div[2]/div/div/div'))
)

```

7. Seguido de esto, el código entrará dentro de un nuevo ciclo, esto para solo entrar a las páginas donde desea extraer los datos. Al entrar busca y elimina el botón de la ventana emergente. Y guardará todos los elementos que poseen un link.

```

for i in range(nItems):
    boton = WebDriverWait(driver,10).until(
        EC.presence_of_element_located((By.XPATH, '//*[@id="Modalveda"]/div/div/div[1]/button/span'))
    )
    boton.click()

    WebDriverWait(driver,10).until(
        EC.element_to_be_clickable((By.XPATH, '//div[@class="container container-small"]/div/div/div/div'))
    ))
    sleep(random.uniform(10.0, 12.0))
    elementos= driver.find_elements_by_xpath('//div[@class="container container-small"]/div/div/div/div/

```

8. Usando la lista que almacena los elementos (links), procedemos a entrar en cada página, e iniciamos con la extracción de los datos. Debido a que cada página tiene expresiones XPath diferentes, cada una de las páginas las colocamos dentro de un if, estos if a su vez contienen otro if que permite realizar las elecciones del usuario, por si ha pedido, los títulos, conceptos. Que es prácticamente lo único que la pagina tenia, además de una gran variedad de pdf's. Al final de if principal, colocamos un driver.back() para poder regresar a la página donde se encuentra nuestro siguiente elemento.

```
try:
    elemento= elementos[i].get_attribute('href')
    driver.get(elemento)
    sleep(random.uniform(10.0, 12.0))
    if elementos[i]==elementos[0] or elementos[i]==elementos[3] or elementos[i]==elementos[8]:
        if xpath1=="titulos" or xpath2=="titulos" or xpath3=="titulos" or xpath4=="titulos":
            titulo2=driver.find_element_by_xpath('/html/body/div[2]/div/div/h2').text
            titulo2= titulo2.replace(',',' ')
            DE.write(titulo2)
            DE.write("\n")
        if xpath1=="conceptos" or xpath2=="conceptos" or xpath3=="conceptos" or xpath4=="conceptos":
            parrafo2=driver.find_element_by_xpath('/html/body/div[2]/div/div/div/div/p').text
            parrafo2=parrafo2.replace(',',' ')
            DE.write(parrafo2)
            DE.write("\n")
    driver.back()
```

```
sleep(random.uniform(10.0, 12.0))
if elementos[i]== elementos[1] or elementos[i]==elementos[2]:
    titulos=driver.find_elements_by_xpath('//div[@class="container landing-wrapper"]/section/headers')
    parrafos=driver.find_elements_by_xpath('//div[@class="container landing-wrapper"]/section/div')
    for k in range(len(titulos)):
        if xpath1=="titulos" or xpath2=="titulos" or xpath3=="titulos" or xpath4=="titulos":
            titulo2=titulos[k].text
            titulo2= titulo2.replace(',',' ')
            DE.write(titulo2)
            DE.write("\n")
        if xpath1=="conceptos" or xpath2=="conceptos" or xpath3=="conceptos" or xpath4=="conceptos":
            parrafo2=parrafos[k].text
            parrafo2=parrafo2.replace(',',' ')
            DE.write(parrafo2)
            DE.write("\n")
    driver.back()
```

```
DE.write("\n")
if xpath1=="conceptos" or xpath2=="conceptos" or xpath3=="conceptos" or xpath4=="conceptos":
    parrafo2=driver.find_element_by_xpath('/html/body/div[2]/div/div/div/div/div/div/p').text
    parrafo2=parrafo2.replace(',',' ')
    DE.write(parrafo2)
    DE.write("\n")
driver.back()
sleep(random.uniform(10.0, 12.0))
if elementos[i]== elementos[6] or elementos[i]== elementos[7]:
    if xpath1=="titulos" or xpath2=="titulos" or xpath3=="titulos" or xpath4=="titulos":
        titulo2=driver.find_element_by_xpath('/html/body/div[2]/div/div/div/div/div/div/h2').text
        titulo2= titulo2.replace(',',' ')
        DE.write(titulo2)
        DE.write("\n")
    if xpath1=="conceptos" or xpath2=="conceptos" or xpath3=="conceptos" or xpath4=="conceptos":
        parrafo2=driver.find_element_by_xpath('/html/body/div[2]/div/div/div/div/div/div/div/div/p').text
        parrafo2=parrafo2.replace(',',' ')
        DE.write(parrafo2)
```

```

if elementos[i]== elementos[6] or elementos[i]== elementos[7]:
    if xpath1=="titulos" or xpath2=="titulos" or xpath3=="titulos" or xpath4=="titulos":
        titulo2=driver.find_element_by_xpath('/html/body/div[2]/div/div/section/h2').text
        titulo2= titulo2.replace(',',' ')
        DE.write(titulo2)
        DE.write("\n")
    if xpath1=="conceptos" or xpath2=="conceptos" or xpath3=="conceptos" or xpath4=="conceptos":
        parrafo2=driver.find_element_by_xpath('/html/body/div[2]/div/div/section/div[2]/div/p').text
        parrafo2=parrafo2.replace(',',' ')
        DE.write(parrafo2)
        DE.write("\n")
driver.back()

```

9. Pasamos a la parte del “except” del try del inicio del código de extracción, en esta sección detendremos el código si es que se encuentra un error durante su ejecución, asimismo se mandará una página HTML que advertirá al usuario de esto. En caso de que el código corra adecuadamente, se terminará la ejecución, se detendrá el controlador del navegador y se cerrará el archivo donde se capturó la data, por último retornara un documento HTML indicando esto y permitiendo descargar el archivo previamente generado.

```

except Exception as e:
    print(e)
    driver.back()
WebDriverWait(driver,6).until(
    EC.presence_of_all_elements_located((By.XPATH, '//div[@class="container container-small"]/div/div/div[2]'))
)
sleep(random.uniform(10.0, 12.0))

except Exception as e:
    print(e)
    print('Error en la extraccion')
    return render_template('ErrorExtraccion.html')
driver.close()
print('Fin de la extraccion')
DE.close()
return render_template('Extraccion.html', doc=datos, dat=documento)

```

10. Por último el “if \_\_name\_\_” que nos sirve para mantener al sitio web actualizado mientras se generan cambios en el código de python, esto durante las pruebas.

```

if __name__=='__main__':
    app.run(debug=True)

```

## ❖ Sitio Web: Características Generales

El sitio web generado para los tres códigos de extracción es prácticamente el mismo, únicamente cambiando un poco las opciones de datos a extraer en los formularios de selección. Por esto mismo, en las tres páginas principales, se implementó un pop up con información básica de uso para el usuario, esto para facilitar su uso aunque no se sepa mucho del tema ni se esté familiarizado demasiado con la tecnología.

-Información de uso-

lección-  
e dato a extraer:  
lección-  
e dato a extraer:  
lección-  
ems a extraer :  
archivo a genera  
lección-  
Extraer d

=Pasos para usar el sitio=

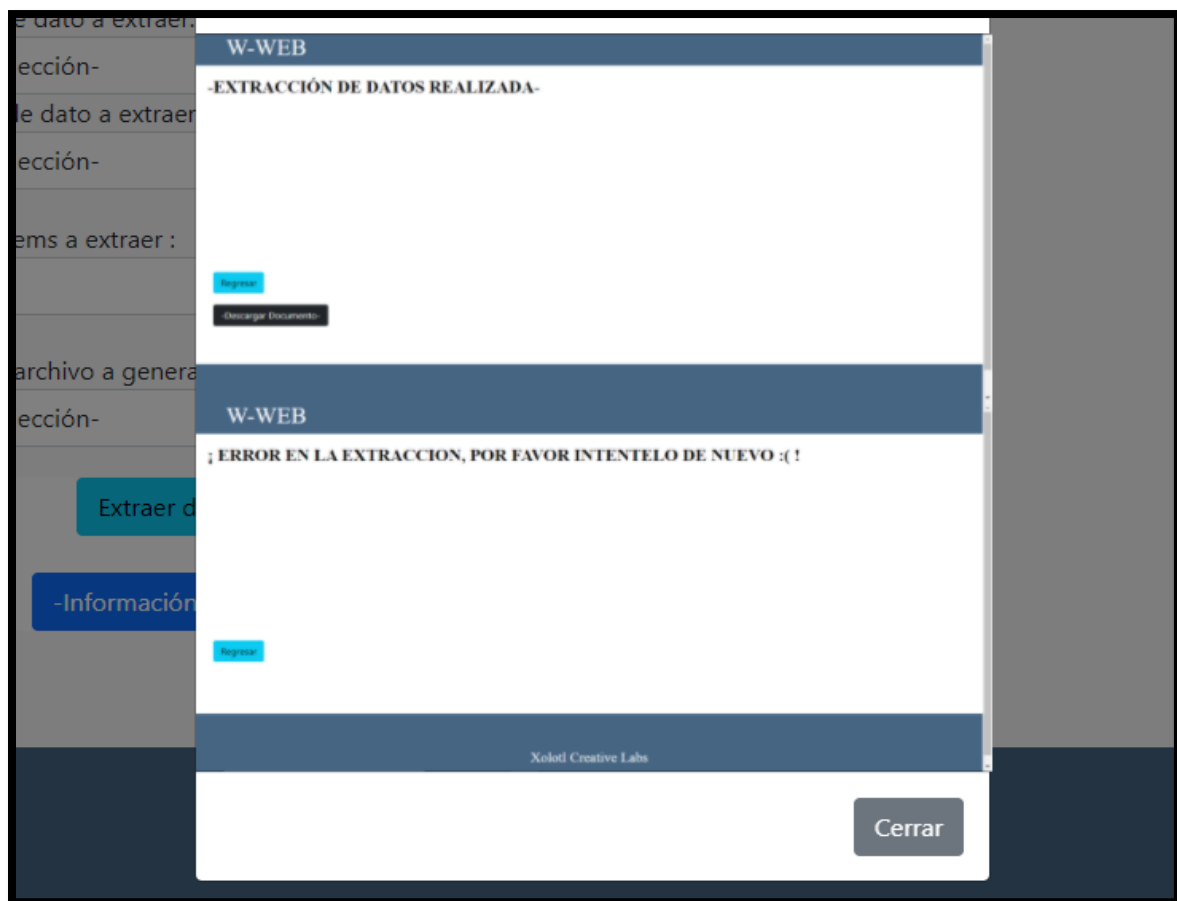
Para poder extraer datos de sitios web desde esta página debes considerar la siguiente información.

Si buscas extraer datos de un sitio web sigue los siguientes pasos:

1. Introduce la URL completa del sitio del cual quieres extraer la información, puedes copiarla desde el navegador
2. Elige los datos a extraer de la página, se te desplegará una lista de posibilidades en cada una de las cuatro barras de selección

on-  
ato a extraer  
ón-  
a extraer :  
ivo a genera  
ón-  
Extraer d

3. En la parte del número de items, debes de ingresar el total de conjuntos datos que quieres extraer, por ejemplo si hablamos de tablas de datos, cada una de una zona en específico, la cantidad de items sería el número total de zonas que se desean extraer, cada una con sus tablas de datos correspondientes
4. Por último elige un formato en el que desees generar el archivo donde se guardará la información obtenida, y clic en "Extraer datos"
5. La página te indicará si la extracción fue exitosa o si hubo algún error, si lo último pasa, regresa e introduce de nuevo los datos, en caso de que persista, lo más probable es que el sitio web no sea válido o este caído

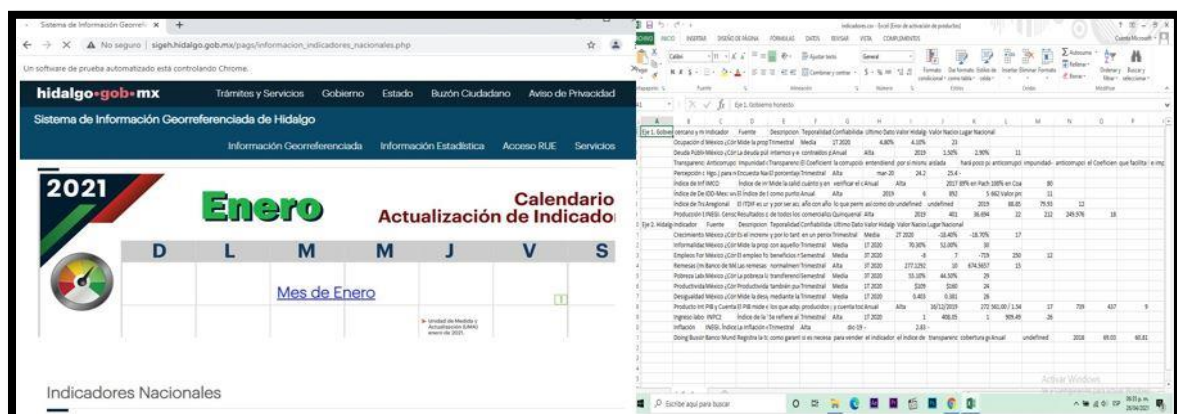


## -Serie de Pruebas-

- ❖ Link : Pagina del SIGEH - Indicadores

### Prueba 1 (segunda versión)

- Tiempo: 11 minutos 45 segundos
- Cantidad de ejes: 1 y parte del segundo.

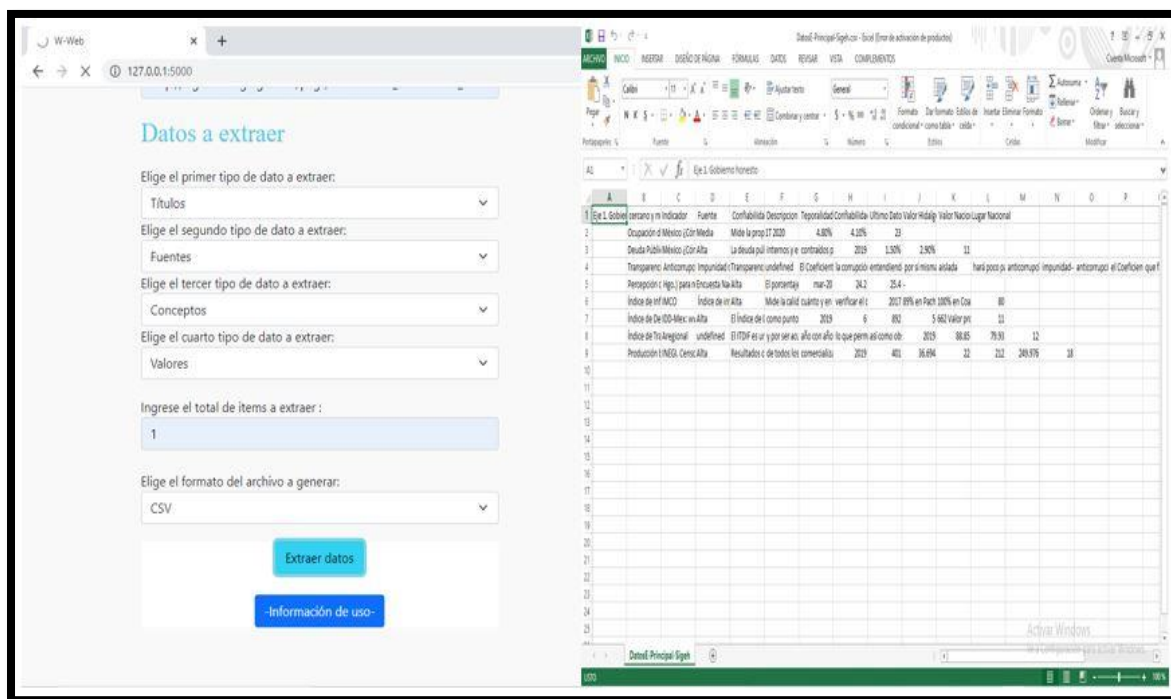




### Pruebas desde el sitio web:

- Tiempo: 3 minutos 24 segundos
- Cantidad de datos extraídos: 1 eje

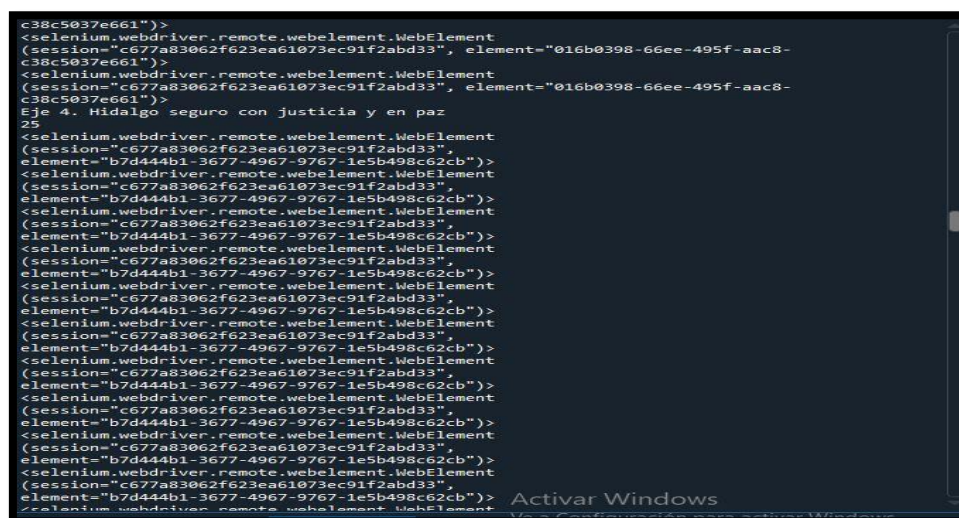
Este código está diseñado para extraer los datos por ejes, así que todos los datos que se extraen son los datos que pertenezcan a ese eje, en este caso solo fue el primero, y se subdivide en 8 indicadores.



❖ Link : [Pagina del SIGEH - Consulta de datos](#)

## Pruebas desde el código:

Al realizar el código, obtuvimos diferentes resultados, por ejemplo el siguiente resultado de una de nuestras pruebas, este resultado fue debido a la falta de colocar `.text` después de pedirle a nuestro código que busque una expresión y luego lo convierta en texto, lo único que está haciendo es imprimir la dirección que encontró por medio de selenium.



El siguiente resultado, es una prueba, donde el código solo nos imprimió una coma(,) y una palabra, seguido de Error. “Una Prueba de Error”.

```
In [4]: runfile('C:/Users/Personal/Desktop/Web Scraping 1/Pruebas Py/WS1.7.py',
wdir='C:/Users/Personal/Desktop/Web Scraping 1/Pruebas Py')
,2 Establecimientos

Error

In [5]: |
```

Después de un par de errores, por fin logramos obtener los datos tanto de conceptos como de valores. Sin embargo, la extracción de datos no es impresa con un formato de tabla, o algo más cómodo para poder observar los datos.

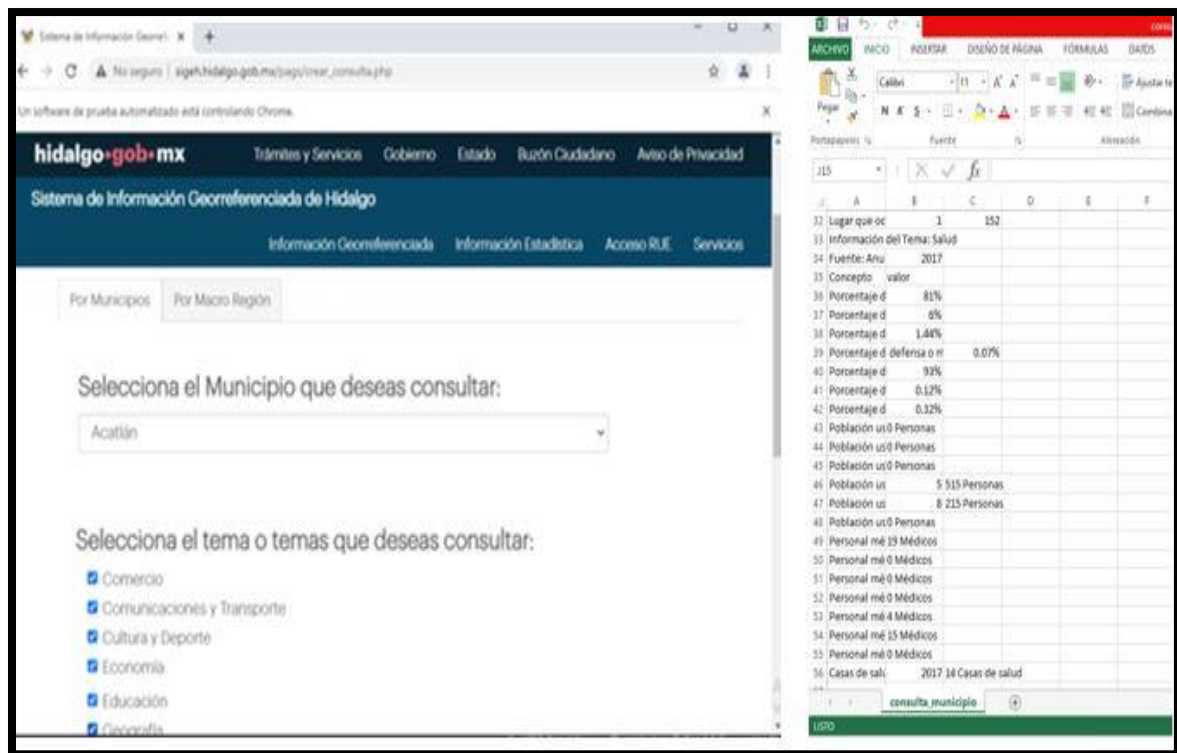
Tiempo de respuesta: 4 minutos 54 segundos 47 mil segundos.

```
Concepto:
Unidades económicas
Personal ocupado dependiente de la razón social
Remuneraciones (Millones de pesos)
Producción bruta total (Millones de pesos)
Valor agregado censal bruto (Millones de pesos)
Total de activos fijos (Millones de pesos)
Población total de 12 años y más
Población Económicamente Activa Total
Población Económicamente Activa Desocupada
Población Económicamente Activa Ocupada
Población no económicamente activa
Población ocupada
Funcionarios, profesionistas, técnicos y administrativos
Trabajadores agropecuarios
Trabajadores en la industria
Comerciantes y trabajadores en servicios diversos
Trabajadores asalariados
Trabajadores no asalariados
Ingreso por trabajo hasta 1 s.m.2
Ingreso por trabajo más de 2 s.m.
Ingreso por trabajo no especificado
Ingreso por remesas Ene-Mar 2018
Ingreso por remesas Abr-Jun 2018
Ingreso por remesas Jul-Sep 2018
Índice de Intensidad Migratoria_2010
Grado de Intensidad Migratoria_2010
valor:
236 Unidades
381 Personas
$ 2.40 Mdp
$ 40 Mdp
$ 21.51 Mdp
$ 21.94 Mdp
0 Personas
45.91%
97.11%
2.88%
54%
0 Personas
```

Activar Windows  
Ve a Configuración para activar Windows.

Prueba con el código de extracción terminado (sin modificaciones, primera versión)  
Este primer código solo extrae los datos, pero aún no está unido al sitio web.

- Tiempo: 2 minutos 49 segundos
- Cantidad de datos extraídos: 1
- Tiempo: 4 minutos 47 segundos
- Cantidad de datos extraídos: 2
- Tiempo: 6 minutos 28 segundos
- Cantidad de datos extraídos: 3



Pruebas desde el sitio web:

- Prueba 1 desde el Sitio Web:
- Tiempo: 2 minutos 49 segundos
- Cantidad de datos extraídos: 3

**URL Semilla**

Ingrese la URL:

**Datos a extraer**

Elige el primer tipo de dato a extraer:

Titulos

Elige el segundo tipo de dato a extraer:

Conceptos

Elige el tercer tipo de dato a extraer:

Valores

Elige el cuarto tipo de dato a extraer:

Fuentes

Ingrese el total de ítems a extraer :

3

Elige el formato del archivo a generar:

CSV

**Extraer datos**

archivo2.csv - Excel (Err)

ARCHIVO INICIO INSERTAR DISEÑO DE PÁGINA FÓRMULAS DATOS REVISAR VISTA COMP

Calibri 11 A A A A Ajustar texto General

Pegar Fuente Alineación Número

A1 Acatlán

1	Acatlán						
2	Información del Tema: Comercio						
3	Fuente: Anuario Estadístico y Geográfico del Estado de Hidalgo. 2017.						
4	Concepto valor						
5	Gasolineras 2 Establecimientos						
6	Tienda Dicor 13 Tiendas						
7	Tianguis 1 Tianguis						
8	Mercados pú 1 Mercados						
9	Centrales de 0 Centrales						
10	Centros de c 0 Centros de Acopio						
11	Liconsa punt 1 Establecimientos						
12	Liconsa fami 88 Familias						
13	Liconsa bené 169 Beneficiarios						
14	Liconsa dota 20304.00 Lt						
15	Liconsa impc \$100.36						
16	Información del Tema: Comunicación y Transporte						
17	Fuente: Anuario Estadístico y Geográfico del Estado de Hidalgo. 2017.						
18	Concepto valor						
19	Total longitu 185.76 Km						
20	Longitud can 0.00 Km						
21	Longitud can 11.20 Km						
22	Longitud can 0.00 Km						
23	Longitud can 0.0 Km						
24	Longitud can 139.82 Km						
25	Longitud can 34.74 Km						

archivo2

- Prueba 2 desde el Sitio Web:
- Tiempo: 1 minuto 55 segundos
- Cantidad de datos extraídos: 1

**URL Semilla**

Ingrese la URL:

**Datos a extraer**

Elige el primer tipo de dato a extraer:

Fuentes

Elige el segundo tipo de dato a extraer:

Valores

Elige el tercer tipo de dato a extraer:

-Abrir menú de selección-

Elige el cuarto tipo de dato a extraer:

-Abrir menú de selección-

Ingrese el total de ítems a extraer :

1

Elige el formato del archivo a generar:

CSV

**Extraer datos**

archivo7.csv - Excel (Err)

ARCHIVO INICIO INSERTAR DISEÑO DE PÁGINA FÓRMULAS DATOS

Calibri 11 A A A A Ajustar texto General

Pegar Fuente Alineación

A1 Acatlán

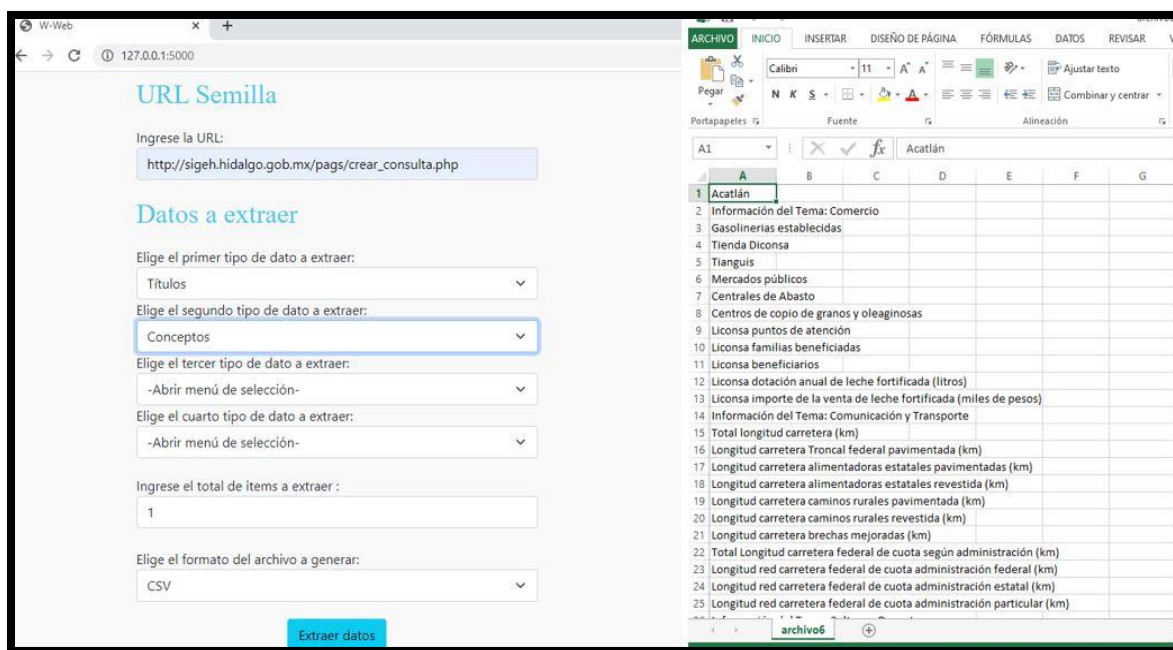
1	Acatlán				
2	Fuente: Anuario Estadístico y Geográfico del Estado de Hidalgo. 2017.				
3	2 Establecimientos				
4	13 Tiendas				
5	1 Tianguis				
6	1 Mercados				
7	0 Centrales				
8	0 Centros de Acopio				
9	1 Establecimientos				
10	88 Familias				
11	169 Beneficiarios				
12	20304.00 Lt				
13	\$100.36				
14	Fuente: Anuario Estadístico y Geográfico del Estado de Hidalgo. 2017.				
15	185.76 Km				
16	0.00 Km				
17	11.20 Km				
18	0.00 Km				
19	0.0 Km				
20	139.82 Km				
21	34.74 Km				
22	0.00 Km				
23	0.00 Km				
24	0.00 Km				
25	0.00 Km				

archivo7



### Prueba 3 desde el sitio Web:

- Tiempo: 1 minutos 46 segundos
- Cantidad de datos extraídos: 1
- Extraer datos de títulos y conceptos desde la página SIGEH, en el apartado crear consulta.



### Prueba 4 desde el sitio Web:

- Tiempo: 0 minutos 1 segundo 45 mil segundos.
- Extraer datos desde una página diferente a la de SIGEH.



❖ Link : Página del gobierno del estado de Hidalgo - Población

### Prueba primera versión terminada:

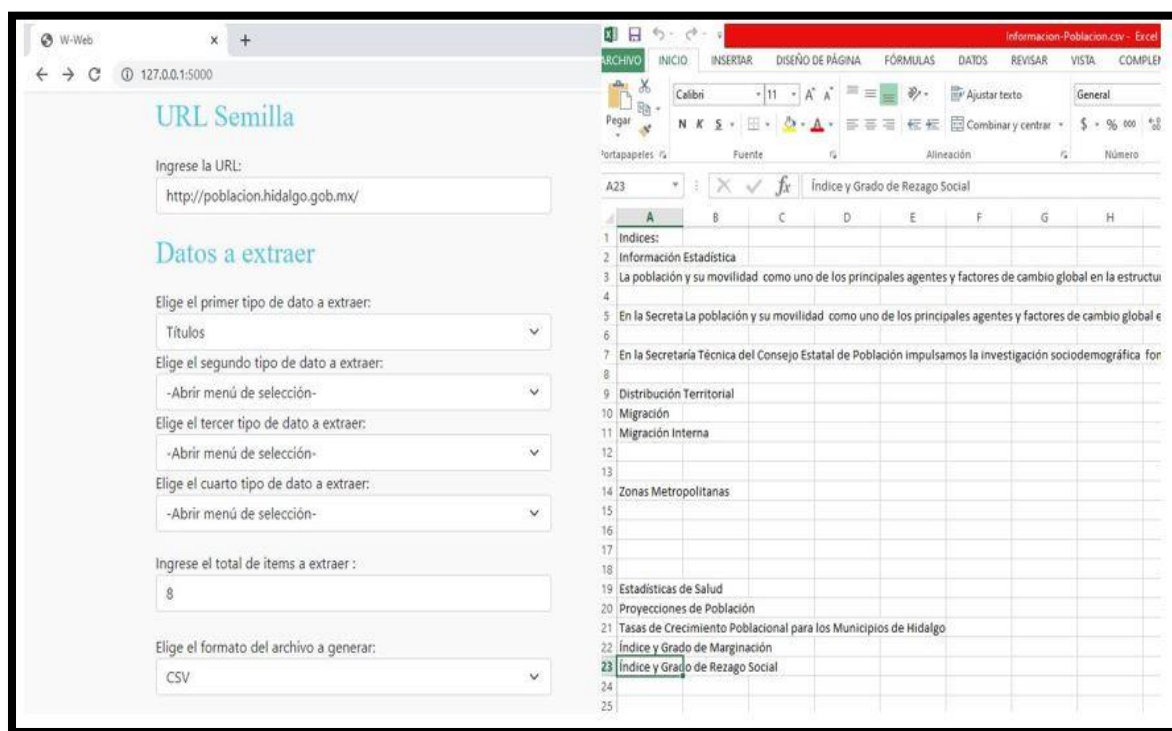
- Tiempo: 2 minutos 36 segundos
- Cantidad de datos extraídos: 2
- Tiempo: 3 minutos 22 segundos
- Cantidad de datos extraídos: 3
- Tiempo: 4 minutos 26 segundos
- Cantidad de datos extraídos: 6
- Tiempo: 5 minutos 21 segundos
- Cantidad de datos extraídos: 7
- Tipos: Títulos, Contextos e indicadores

The screenshot displays the 'hidalgo.gob.mx' website interface. The top navigation bar includes links for 'Gobierno', 'Trámites y Servicios', 'Hidalgo Travel', 'Transparencia', and 'Buñón Ciudadano'. The main content area is titled 'Consejo Estatal de Población' and 'Información Estadística'. It contains a paragraph about population and mobility, followed by a section with three buttons: 'Distribución Territorial', 'Migración', and 'Zonas Metropolitanas'. Below these are three more buttons: 'Estadísticas de Salud', 'Proyecciones de Población', and 'Información Sociodemográfica'. To the right of the website is a Microsoft Excel spreadsheet titled 'Poblacion.xlsx - Excel (Sistema de extracción de productos)'. The spreadsheet has columns A through J and rows 1 through 25. It contains data related to population statistics, including a table with headers 'Indicadores', 'Población', and 'Crecimiento'. The data includes values for population density, life expectancy, and migration statistics.

### Prueba 2 (primera versión):

- Tiempo: 6 minutos 24 segundos
- Cantidad de datos extraídos: 8
- Tipo de datos: Título
- En esta extracción de datos, solo se usó como prueba, para asegurarnos de que al solo pedir títulos, solo nos arroje los títulos.





## ***-Conclusiones-***

Se consiguió realizar el web scraping de tres páginas diferentes usando el framework de Selenium junto con el lenguaje de programación Python, y aunque pudo haberse hecho con un solo código, el tiempo en el que se desarrolló y los conocimientos que se necesitaban fueron los limitantes que impidieron que sucediera. Sin embargo se consiguió cumplir con lo que se necesitaba en el ámbito funcional, y además se agregaron algunos detalles en el campo de la interfaz de usuario que sirvieron como un extra que facilita el uso del proyecto para la extracción de la información.

## ***-Observaciones-***

Hubo problemas al momento de extraer la información del sitio de la población del estado de Hidalgo dado que la página en ocasiones respondía adecuadamente y en otras no a pesar de que se usaba el mismo código.

Se buscó la forma de procesar la data con la librería "Pandas" de Python, sin embargo el tiempo y el hecho de que se guardaba la información en archivos de diferente formato impidió que se pudiera lograr.

También se planeaba generar el archivo donde se guardaba la data como un descargable desde la página web, pero hubo problemas en el testeó, así que no se ha probado mucho su funcionamiento, aunque se espera que funcione una vez que se aloje en un host.