

BOT - Automatize



Centro de Estudios Superiores (CES) Afuera
Ciclo Formativo de Grado Superior (CFGS) de
Desarrollo de Aplicaciones Multiplataforma con
Big Data (DAM)

Trabajo Fin de Grado (TFG)

Trabajo realizado por: Eduardo

ÍNDICE

1.- Abstract.....	3
2.- Objetivos.....	4
3.- Arquitectura.....	5
4.- Recursos utilizados.....	11
5.- Instrucciones de uso.....	12
6.- Base de datos.....	15
7.- Futuras mejoras.....	15
8.- Conclusiones.....	17
8.- Enlace repositorio github.....	19
9.- Bibliografía.....	19
10.- Agradecimientos.....	20

1.- Abstract

El proyecto se basa en la creación de una herramienta dedicada a automatizar los clicks en un botón concreto de los sitios web, acelerando significativamente las tareas repetitivas que requieren interacción con los sitios web en cuestión.

Simplemente cargando un archivo de Excel que contiene enlaces, nuestro sistema abre estas páginas y hace click en el botones indicado, sin necesidad de que los usuarios lo tengan que realizar de forma manual, realizando al final de la ejecución un informe detallado que permiten un análisis meticuloso de las interacciones realizadas. Con el uso de esta herramienta se prevé una reducción significativa en la cantidad de mano de obra requerida para realizar tareas repetitivas en sitios web. Con menos tiempo dedicado a estas tareas repetitivas, los usuarios pueden centrar su atención en actividades con más valor. Está claro que esta solución práctica ofrece a las empresas una oportunidad excepcional para mejorar su productividad.

The project is based on the creation of a tool dedicated to automating clicks on a specific button on websites, significantly speeding up repetitive tasks that require interaction with the websites in question. By simply uploading an Excel file that contains links, our system opens these pages and clicks on the indicated buttons, without the need for users to have to do it manually, making a detailed report at the end of the execution that allows analysis. meticulous of the interactions carried out. With the use of this tool, a significant reduction in the amount of labor required to perform repetitive tasks on websites is expected. With less time spent on these repetitive tasks, users can focus their attention on higher value activities. It is clear that this practical solution offers companies an exceptional opportunity to improve their productivity.

2.- Objetivos

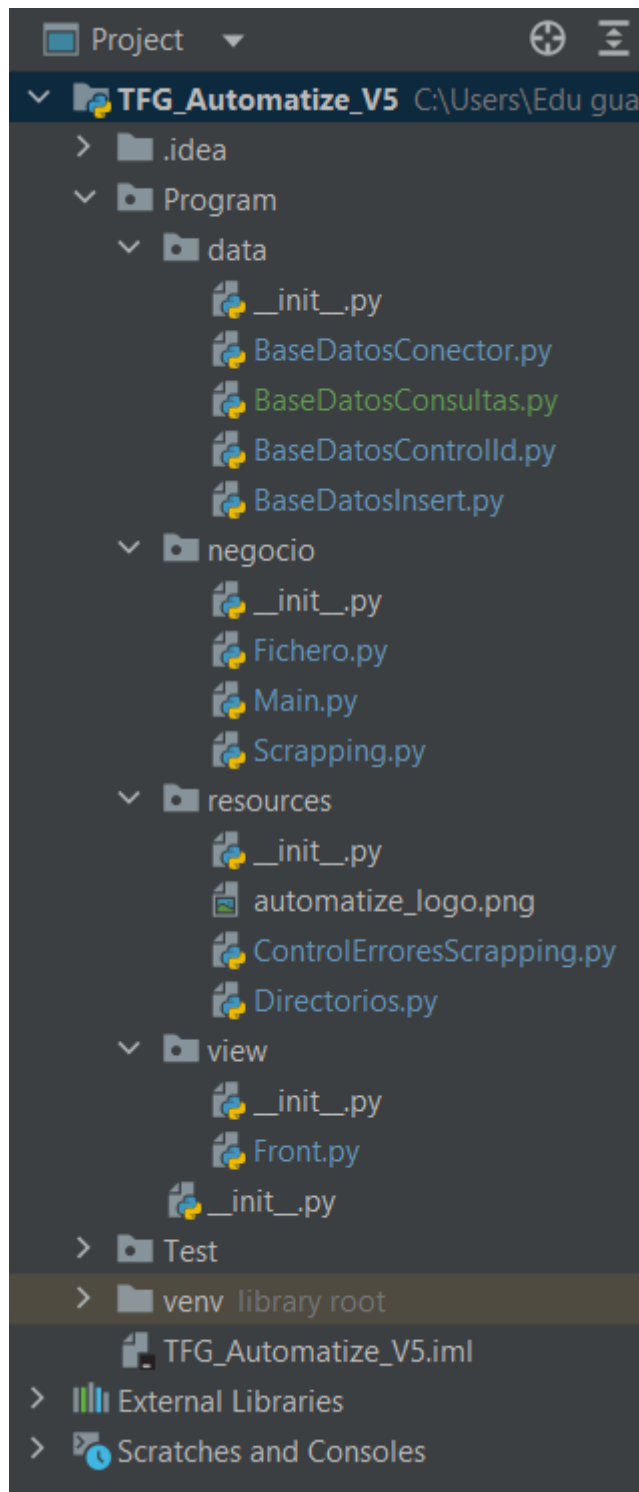
El objetivo del proyecto es la creación de un programa .exe que pueda cargar un documento .XLS que contenga un número aleatorio de links, teniendo que tener estos links el mismo host.

También deberá de tener un apartado para poder introducir la búsqueda Xpath del elemento que queremos que sea pulsado.

Al ejecutar el programa, deberá abrir el navegador, cargar el enlace y pulsar el elemento que le hayamos enviado a través de una búsqueda Xpath, después repetirá el proceso hasta hacerlo con todos los enlaces cargados en el documento .XLS

3.- Arquitectura

El proyecto cuenta con 2 directorios principales, uno para los Test que se dejará vacío, ya que de momento no será necesario hacer test y otro directorio Program en el que se encuentran los directorios con el código del programa, ordenado de la siguiente manera:



- **data:** todo lo necesario para trabajar con la base de datos donde se registra el log de la ejecución del programa
 - **BaseDatosConector.py:** aquí se encuentra el conector de la base de datos, con todos los datos necesarios para conectarnos con la misma:

```
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="MySQL",
    database="log_automatize_1"
)
```

El código inicial para crear la tabla es el siguiente:

```
CREATE TABLE nombre_de_la_tabla (
    idNumEjecucion INT,
    ejecucion_moment DATETIME,
    doc_path VARCHAR(255),
    xpath VARCHAR(255),
    urlDoc VARCHAR(255),
    urlActual VARCHAR(255),
    urlPost VARCHAR(255),
    controlErrores VARCHAR(255)
);
```

- **BaseDatosControlId.py:** para conseguir que el ID de cada ejecución que realiza el programa sea consistente, se ha necesitado crear un método que lea el ID del último registro y actualice el ID del nuevo registro.
- **BaseDatosInsert.py:** aquí se realiza el grabado del log por cada vez que el programa carga un enlace y pulsa un botón.

- **negocio:** los métodos principales en los que se basa el programa
 - **Fichero.py:** lectura de un documento .XLS y volcado en un array

```
def readFichToArray(filePath):

    #Reconocimiento del archivo .XLS
    doc = pd.read_excel(filePath, header=None)
    dimension = doc.shape
    tamFilas = dimension[0]
    #Control de lectura de filas
    print(f"tamFilas: ", tamFilas)

    # Acceder a la primera fila
    columna = doc.iloc[:, 0]

    #Para un mejor manejo vuelco el documento en un array
    arrayEnlaces = []
    for x in columna:
        arrayEnlaces.append(x)

    return arrayEnlaces
```

- **Main.py:** ejecución del programa

```
def main():
    # El front tiene que eestarse ejecutando siempre que
    # se abra el programa
    Front.ventana.mainloop()

if __name__ == '__main__':
    main()
    # Control para saber si la bd está bien conectada
    print(BaseDatosConector.mydb)
```

- **Scrapping.py:** bucle que lee el array donde se encuentran los enlaces y por cada elemento, abre el contenido del mismo en un navegador y pulsa un determinado botón. La parte más importante del proyecto:

```
##### IMPORTANTE #####
# Estos argumentos los traigo desde el Front
# BungaEdu *
def doScrappingWeb(idNumEjecuciones, columna, busquedaXpath, filePath):
    # Conecto con el driver para poder trabajar con Chrome
    driver = webdriver.Chrome()

    # Bucle de lectura-ejecución por cada enlace
    for enlace in columna:
        # Recojo un enlace
        driver.get(enlace)
        urlDoc = "URL_doc: " + enlace
        print(urlDoc)

    # Para un mejor funcionamiento maximizo la pestaña
    driver.maximize_window()

    # Para control, reviso si se ha abierto en browser el mismo enlace del documento
    urlActual = driver.current_url
    print("URL_init: " + urlActual)

    # Realizo una espera hasta que detecto que mi botón está cargado, si hay overtime
    # el programa se parará
    wait = WebDriverWait(driver, 10)
    button = wait.until(EC.presence_of_element_located((By.XPATH, busquedaXpath)))

    # Realizo otra espera para que el programa no se pare, para darle tiempo después de
    # encontrar el botón
    time.sleep(2)
    button.click()

    #Realizo una comprobación para ver si la URL después de pulsar el botón es igual o no
    urlPost = driver.current_url
    print("URL_post: " + urlPost)
    time.sleep(2)
    controlErrores = ControlErroresScrapping.controlErrores(urlActual, urlPost)

    # Grabo toda la información de los pasos que realiza la aplicación
    BaseDatosInsert.insertarRegistro(idNumEjecuciones, datetime.now(), filePath, busquedaXpath,
                                     urlDoc, urlActual, urlPost, controlErrores)

    driver.quit()
```


- **resources:**

- **automatize_logo.png:** imagen del logo de la aplicación
- **ControlErroresScrapping.py:** aquí se analiza el trabajo de la aplicación y según lo que haya ocurrido durante la ejecución, se registrará en el log.
- **Directorios.py:** para que la aplicación pueda exportarse, la ruta principal tiene que ser la del proyecto, por ello este método devolverá siempre la raíz del proyecto.

- **view:**

- **Front.py:** aquí se encuentra el código que crea la interfaz del usuario y también se encuentra el código que realiza funciones en base de la interacción entre el usuario y el programa.
 - Aquí tenemos la visualización del programa, la parte de la interfaz:

```
# VARIABLES GLOBALES
# Tenemos que crear variables globales ya que
# el programa se ejecuta con estas variables vacías
filePath = ""
busquedaXPath = ""

# VENTANA PRINCIPAL
# Creación de la ventana donde vamos a visualizar la aplicación
ventana = tk.Tk()
ventana.geometry("800x600")
ventana.configure(bg='FFFFFF')
ventana.title(" ")

# IMAGEN LOGO
imagenLogo = tk.PhotoImage(
    file=Directorios.rutaRepositorio() + "\\Program\\resources\\automatize_logo.png")
cajaImagen = tk.Label(ventana, image=imagenLogo)
cajaImagen.pack()
cajaImagen.place(relx=0.5, rely=0.15, anchor="center")
cajaImagen.configure(bg='FFFFFF')
```

- Aquí tenemos el botón de examinar, que al seleccionarlo accedes directamente al sistema de archivos:

```
# BOTÓN EXAMINAR DOCUMENTO
# BungaEdu *
def open_file():
    global filePath
    # Recogemos el path del documento
    filePath = filedialog.askopenfilename()
    cajaNombreDocumento.insert(0, filePath)
    print(filePath)

# Cuando pulsamos el botón examinar se ejecuta la función open_file
botonExaminar = tk.Button(ventana, text="Examinar", command=open_file)
botonExaminar.pack()
botonExaminar.place(x="600", y="200", width="100")
```

- Esta es la segunda parte más importante del programa, cuando pulsamos el botón de Ejecutar, se procesa el programa de scrapping y se recoge toda la información que introduce el usuario, necesaria para que se ejecute la parte de scrapping.

```
# Cuando se ejecuta el programa, se recoge la información en esta caja
new *
def inputBusquedaXpath():
    return cajaBusquedaXpath.get()

# Proceso que se realiza al pulsar el botón ejecutar
# BungaEdu *
def ejecutar():
    global filePath
    global busquedaXpath
    # Recogida información caja Búsqueda Xpath
    busquedaXpath = inputBusquedaXpath()
    print(busquedaXpath)
    # Análisis sobre el ID del número de ejecución (primary key)
    idNumEjecucion = BaseDatosControlId.controlId()
    print("Número de registros: " + str(idNumEjecucion))
    # Ejecución del programa de scrapping
    Scrapping.doScrappingWeb(idNumEjecucion,
                             Fichero.readFichToArray(filePath), busquedaXpath, filePath)

# BOTÓN EJECUTAR
botonEjecutar = tk.Button(ventana, text="Ejecutar programa", command=ejecutar)
botonEjecutar.pack()
botonEjecutar.place(relx=0.5, rely=0.5, anchor="center", y="100")
botonEjecutar.configure(bg='#28df28')
```

4.- Recursos utilizados

- **IDE principal:**

IntelliJ IDEA 2022.2.5 (Community Edition), PythonCore (222.4554.5)

- **Lenguaje:** Python 3.9.0

- **Librerías utilizadas:**

- mysql.connector: para utilizar la base de datos.
- pandas: para lectura de fichero .XLS
- selenium: para trabajar con las webs
- time: para poder poner pausas mientras carga el programa
- datetime: calcular el timestamp
- telnetlib: para poder trabajar con búsquedas Xpath
- os: para poder contactar con el sistema interno de archivos
- tkinter: para poder crear la interfaz de la aplicación

- **Descargas necesarias:**

- pip install mysql
- pip install pandas
- pip install xlrd: para poder trabajar con documentos .XLS
- pip install selenium
- Chrome Driver: <https://chromedriver.chromium.org/>

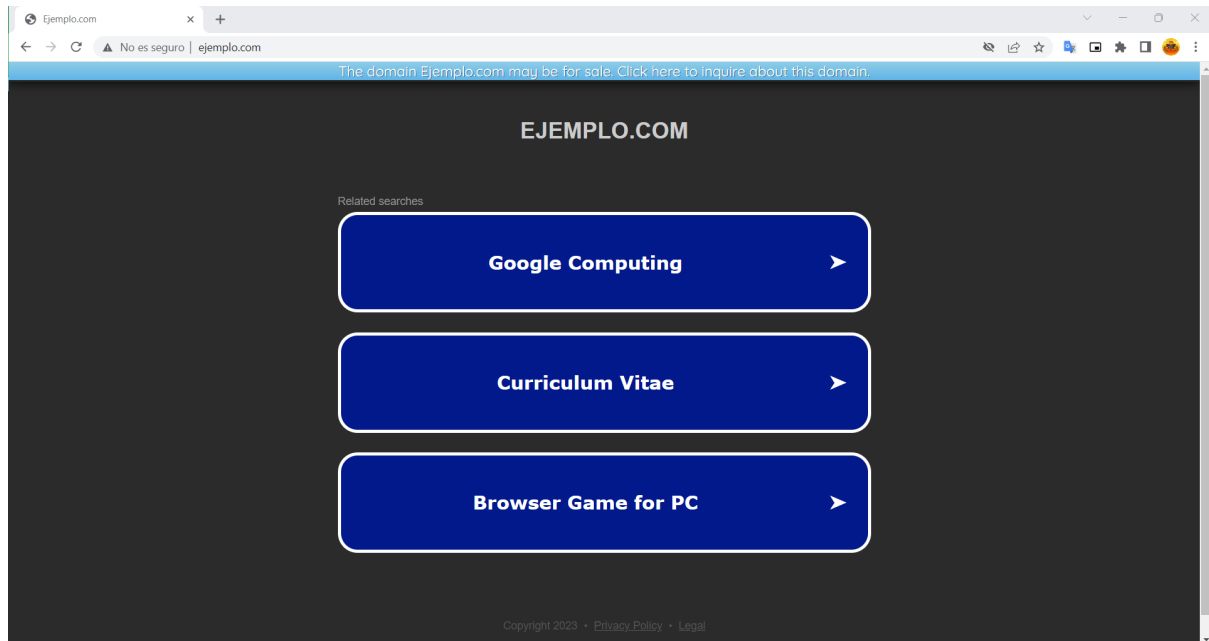
5.- Instrucciones de uso

Para que el usuario pueda utilizar la herramienta, tiene que cumplir las siguientes condiciones de uso:

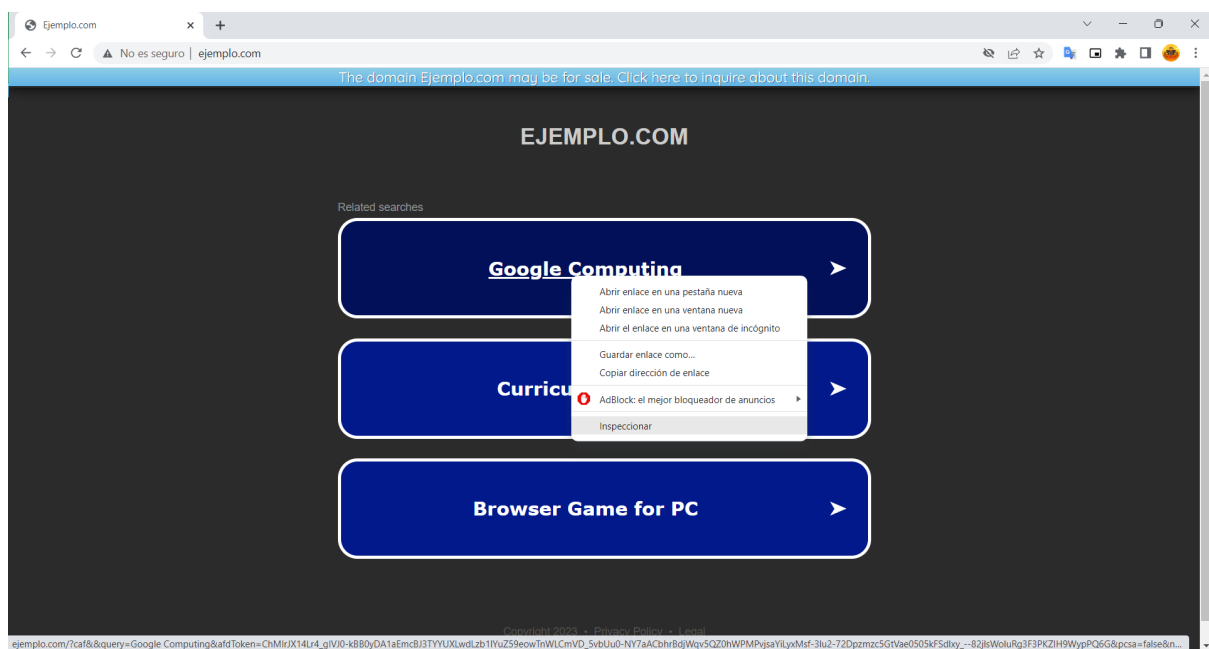
- **Documento .XLS:** en un documento .XLS se tiene que guardar un enlace por cada celda y tendrán que ponerse todos los enlaces en la columna A.
- **Misma raíz en todos los enlaces:** los enlaces que se carguen en la aplicación, tienen que ser del mismo host. Lo único que puede cambiar es la especificación, ya que utilizaremos el mismo botón y por seguridad, necesitamos que la estructura de la página no cambie.
- **Control anti-scraping:** la mayoría de páginas tienen bloqueadores de scraping, por lo que cuando detectan que se está accediendo en navegador oculto o a través de un software automatizado, bloquea la pantalla con algún tipo de banner. Por lo que tienes que verificar abriendo la página en un navegador oculto, si aparece algún banner o no, en el caso de que aparezca, no será posible utilizar el programa con ese host.
- **Cargar el documento:** en la aplicación, seleccionar el botón de Examinar y seleccionar el documento en el que se ha cargado previamente los enlaces. Es importante saber que sólo se puede introducir la ruta a través del botón de examinar, en el caso de que se ponga de forma manual, dará error.

- **Instrucciones para copiar la búsqueda Xpath del botón:**

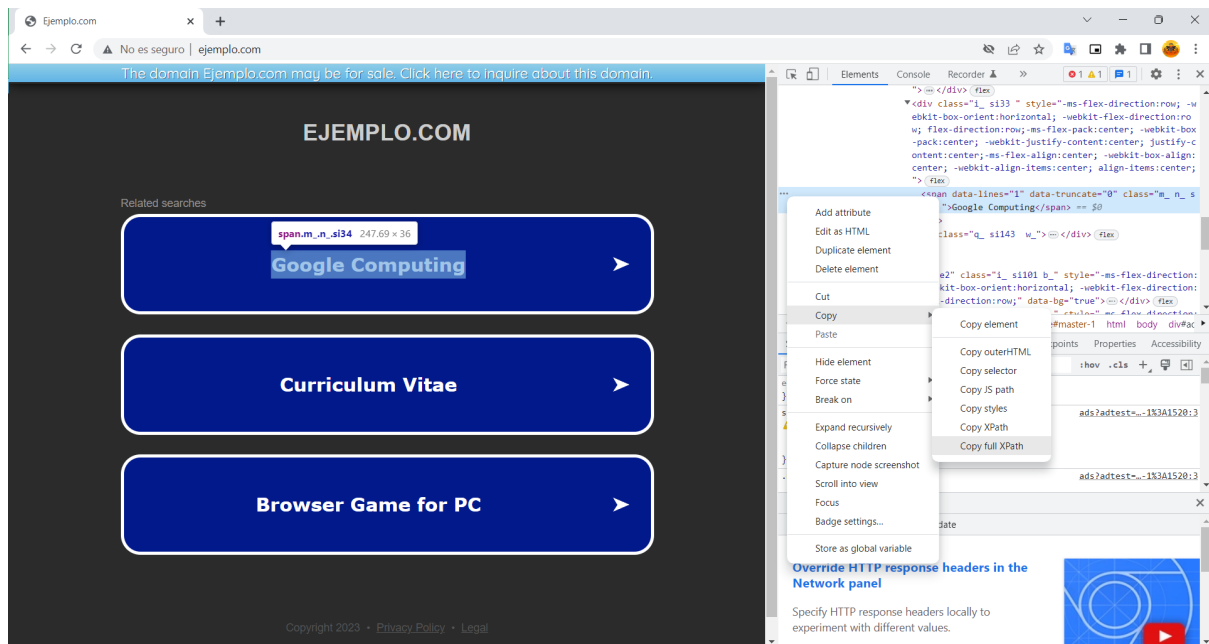
- Abrir el host en Chrome



- Seleccionar el botón con click derecho y en el desplegable seleccionar la opción de inspeccionar.



- En el inspeccionador aparecerá una fila azul, haz click derecho sobre ella, en el desplegable selecciona la opción de copy y en el siguiente desplegable, selecciona la opción de Copy full XPath.



- **Ejecución del programa:** al pulsar el botón de Ejecutar, se abrirá el navegador de Chrome y empezará a realizar la acción de cargar un enlace, pulsar, luego el siguiente y así hasta que no haya más información en el documento.

6.- Base de datos

De cara al administrador del programa, se ha creado una base de datos MySQL con una sólo tabla con el registro de un log, en este caso en root, donde se registra cada enlace del documento que ha sido leído, abierto en el navegador y pulsado, con un enlace resultante.

Los campos de esta base de datos son los siguientes:

- idNumEjecucion: el ID se suma cada vez que se pulsa el botón de Ejecutar.
- execution_moment: el timestamp que se registra cada vez que se realiza el proceso de abrir enlace, pulsar y leer el url resultante del pulsado.
- doc_path: cada línea del documento cargado.
- xpath: búsqueda Xpath introducida.
- enlace_from_doc: enlace leído del documento.
- enlace_from_browser_init: enlace que se ha cargado en el browser.
- enlace_from_browser_post: enlace que se ha generado después de pulsar el botón.
- enlace_from_browser_result: diferencia entre el link inicial y el link después de pulsar

el botón, en el caso de que sean diferentes se grabará en el registro “OK” y en caso contrario “Ha ocurrido un error”.

Diagrama MER de la base de datos:

log_automatize_1
log
idNumEjecucion (PK)
execution_moment
doc_path
xpath
enlace_from_doc
enlace_from_browser_init
enlace_from_browser_post
enlace_from_browser_result

7.- Futuras mejoras

Para próximas actualizaciones se trabajará en la versatilidad del programa. Se podrán cargar las extensiones de documento excel más utilizadas (.xls, .ods, .xlsx) y

también se podrá elegir la opción de cargar documentos online tanto de Microsoft 365 como de Google Drive.

Es necesario también hacer muchos controles de errores, introducir muchos try-catch para que siempre se ejecute el programa, pero que quede grabado en la base de datos lo ocurrido.

También se añadirá funcionalidades al click, es decir, desde la página web del documento, se selecciona un botón y en la siguiente página web, se puede pulsar otro botón.

En una siguiente fase del programa, se crearán versiones diferentes para cada uso, así por ejemplo, se puede hacer un programa para utilizar en una web concreta y pulsar varios botones, introducir textos en barras de búsqueda, interactuar con la web, etc.

En una última fase del programa se actualizará el comportamiento utilizando corrutinas, reduciendo el tiempo de ejecución al máximo.

Igualmente para un mejor funcionamiento y para que el log sea más eficiente, sería necesario poner la base de datos online en AWS.

Como futura mejora ideal, será unir el programa con inteligencia artificial para que la función sirva con todas las páginas, ya que muchas páginas tienen control para el scrapping, pero si el programa es capaz de leer el HTML de la página se podría detectar si tiene algún bloqueador de scrapping y pasarlo.

8.- Conclusiones

El proyecto finalmente cumple con las funcionalidades para las que se creó, es decir, para una actividad concreta de “pulsado en bulk” y sin pocas funcionalidades adicionales.

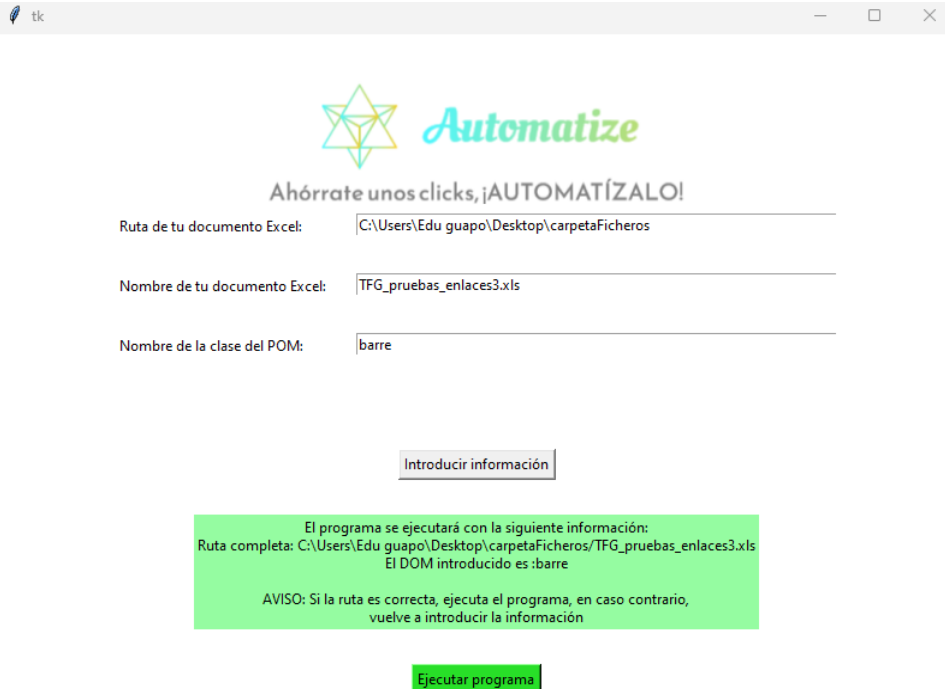
Aparte de ello he conseguido crear una interfaz del programa y dejar el programa en un .exe. También he conseguido ampliar el proyecto y crear un registro del log del funcionamiento de la aplicación en una base de datos.

He aprendido mucho sobre la librería Selenium en diferentes lenguajes (Java & Python) y el comportamiento y funcionamiento del scrapping.


He aprendido sobre la unión del front y el back, a ordenar el código de la aplicación y a entender que no hay límites en la programación, que la frase “si puedes pensarlo, puedes programarlo es totalmente cierta”, ya que durante el proceso de creación, he pasado por varios breackdowns en los que pensaba que mis ideas no eran posibles de realizar.

Por ejemplo, el inicio del proyecto fue lo más tedioso ya que no sabía hasta qué punto Selenium era una librería funcional, ya que estuve 1 mes para que se abriese el navegador y hasta 2 meses sin que se pulsase un botón. Hay que aprender a utilizarla para exprimir todo su potencial.

Después al hacer el front, una primera presentación iba a ser así:



tk

 **Automatize**

Ahórrate unos clicks, ¡AUTOMATÍZALO!

Ruta de tu documento Excel: C:\Users\Edu guapo\Desktop\carpetaFicheros

Nombre de tu documento Excel: TFG_pruebas_enlaces3.xls

Nombre de la clase del POM: barre

Introducir información

El programa se ejecutará con la siguiente información:
Ruta completa: C:\Users\Edu guapo\Desktop\carpetaFicheros\TFG_pruebas_enlaces3.xls
El DOM introducido es :barre

AVISO: Si la ruta es correcta, ejecuta el programa, en caso contrario, vuelve a introducir la información

Ejecutar programa

En esta presentación el usuario tenía que introducir la ruta a mano y después encontré la posibilidad de añadir el botón de examinar, haciendo todo más fácil. Al principio sólo se iba a poder pulsar botones con clase POM, lo cuál limitaba mucho el funcionamiento. Para solucionar esto encontré que podía utilizar búsquedas Xpath, pero la funcionalidad Xpath estaba deprecated en selenium, con la suerte de encontrar un parche realizado con `expected_conditions`. Estas búsquedas las realizaba a través de ChatGTP, por lo que tendría que incorporar la API, lo cuál iba a ser imposible. Pero finalmente, descubrí que Chrome tiene la funcionalidad de generar un Xpath a través del propio navegador. Al tener que introducir una base de datos, ha sido la parte más apasionante de todas porque no creía que fuese capaz, no sabía por donde empezar, no por el código, sino por el planteamiento del proyecto, ya que no sabía en qué momento ni cómo se tenían que grabar los registros, o cómo crear la tabla y he visto que todo lo que he aprendido en base de datos en general, me ha servido para realizar toda esta parte.

Para concluir, ha sido un trabajo emocionante que ha unido un problema, que fue el culpable de que me empezase a interesar la programación y finalmente he creado una herramienta que va a ser capaz de solucionarlo, por lo que he aprendido que soy capaz de hacer todo lo que me proponga con la misma ilusión con la que empecé con mi primer "Hola mundo".

8.- Repositorio Github

https://github.com/BungaEdu/TFG_Automatize_V5.git

9.- Bibliografía

Aquí tengo el funcionamiento de pandas read excel:

<https://estadisticamente.com/leer-fichero-excel-con-python-usando-pandas/>

Lenguaje para coger columnas y ver el tamaño del doc en pandas:

<https://www.delftstack.com/es/howto/python-pandas/python-pandas-df-size-df-shape-and-df-ndim/>

Abrir enlaces desde python:

<https://micro.recursopython.com/recursos/como-abrir-y-leer-urls.html>

Librería selenium:

<https://aprendepython.es/pypi/scraping/selenium/#interacciones>

Api selenium:

https://www.selenium.dev/selenium/docs/api/py/webdriver_firefox/selenium.webdriver.firefox.options.html#module-selenium.webdriver.firefox.options

Chrome driver:

<https://chromedriver.chromium.org/>

Tkinter:

<https://docs.python.org/es/3/library/tkinter.html>

10.- Agradecimientos

Quiero agradecer primero de todo a mi familia, que fueron los primeros que me animaron a tirarme a una piscina en la que ninguno sabíamos si iba a haber agua y que cuando me tiré han sido los flotadores y salvavidas que he tenido para no ahogarme en ningún momento.

A todos mis compañeros de Just Eat, en especial a mi equipo del antiguo departamento CEX que pusieron los pilares de la disciplina y el orden en mi vida laboral, que ha hecho que pueda concluir satisfactoriamente el curso con buenas notas y afrontar cualquier reto con una gran sonrisa.

Gracias a todos mis amigos, que estuvieron y a los que están, porque sin ellos hubiese sido imposible llegar a realizar mi primera aplicación.

Y por último, pero no por ello menos importantes, a los mentores y guías que he tenido en todo este proceso:

Luis Felipe Vélez Flores: por plantearle el problema que resuelve este TFG y por su respuesta “eso con 2 librerías de Python lo sacas”, cuando ni siquiera sabía qué era un lenguaje de programación.

José Montemayor: por ser el responsable de mi primer “Hola Mundo”, por enviarme un libro de 500 hojas de Python para “motivarme” y por darme algunos minutos de su tiempo para la comida, sólo para hablarme sobre la programación.

Jon Zamora Oyarzun: por toda la ayuda al cambiar el chip de humano a informático, por su interés en mi trabajo y por hacerme ver que podía ser programador.

Carlos Rossique: por toda la paciencia, por enseñarme a estudiar y por la afirmación más importante que he podido escuchar como estudiante y que marcará todo mi proceso de desarrollo como programador: “Eduardo, ninguna pregunta es tonta”.