Automatización de procesos en sitios web y generación de informes, 2023

Ariel Eduardo Bedoya Marín

Informe de práctica para optar al título de Ingeniero de Sistemas

Asesor
Jaime Humberto Fonseca Espinal, Especialista en Ciencias Electrónicas e informática

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas
Medellín, Antioquia, Colombia
2023

Contenido

Siglas, acrónimos y abreviaturas

BPO	Business Process Outsourcing
UdeA	Universidad de Antioquia

Resumen

La práctica académica profesional, la cual se realizó con el objetivo de adquirir experiencia a nivel laboral y profesional en el campo de la ingeniería de sistemas, fue realizada en la empresa Emtelco, en el área de programación y control. Dicha práctica se efectuó realizando y apoyando proyectos de automatización de reportes utilizando el lenguaje de programación Python, Estos proyectos se presentaban periódicamente y de acuerdo con las solicitudes emergentes por parte de los analistas de reportes y así mismo se establecía un periodo de tiempo determinado para su entrega y puesta en ejecución.

Durante el periodo de prácticas se realizó un total de cuatro (4) automatizaciones, de las cuales, las dos primeras fueron proyectos ya existentes o más específicamente ya iniciados, pero en un nivel de desarrollo temprano. Las dos automatizaciones siguientes se realizaron desde su fase inicial, es decir, desde cero. Los cuatro proyectos fueron finalizados de manera satisfactoria siendo esto demostrado en su puesta en ejecución dentro del entorno requerido y así mismo por la verificación del profesional que acompañaba en la realización de las tareas de la práctica profesional.

Es importante mencionar que, durante el último mes de la práctica profesional, se presentó en la empresa un evento de ciberseguridad, el cual conllevó a la necesidad de implementar un plan de contingencia para restablecer las plataformas tecnológicas y los procesos del área y por ende de la empresa. Esta experiencia, aunque inesperada y nada deseada para la empresa y sus intereses, fue enriquecedora en cuanto a conocimiento frente a las adversidades y retos que recaen en los entornos laborales y empresariales.

Palabras clave: lenguaje de programación, práctica académica, python, automatización de reportes, plan de contingencia, ataque informático.

Abstract

The professional practice was made pointing to gain working experience in the field of systems engineering and it was successfully done at Emtelco, in the control and programming area. Such practice was accomplished taking and supporting on projects for report automation using the Python programming language. Those projects were presented in a chronological order and according to the requests from the report analysts. It was also, a timeout for the implementation of the code.

There was a total of four (4) projects during the practice, of which, the first two were existing projects but in early development. The other two automations were created right from the start. All four projects were successfully completed, and it was proven with its implementation in the requested environment and by the approval and acceptance of the engineer who was in charge and who was advising about the practice.

It is important to mention that during the last month of the practice, an extraordinary event happened which was a cyber-attack against the company platforms. This event led to the need of taking action for restart and resume all the affected technological platforms and processes from the area and the company. This experience, though not expected or desired for the business and its interests, was enlightening in terms of knowledge against adversities and challenges in the professional environment.

*Keywords***:** programming language, professional practice, Python, report automation, cyber-attack.

Introducción

Con este informe se pretende dejar evidencia del trabajo realizado durante el periodo de práctica académica profesional, el cual se realizó en la empresa Emtelco, desarrollando proyectos de programación de scripts para la automatización de reportes mediante herramientas utilizadas en el lenguaje Python incluyendo funciones propias para el procesamiento y transformación de la información como la función dataframe y librerías como Pandas. También se empleó una librería denominada *Selenium WebDriver*, para la ejecución de técnicas de *web scraping*, es decir, técnicas para extraer información de sitios web de manera automática. Básicamente el proceso de cada proyecto de automatización se basa en la de extracción de información de determinados sitios web por medio de la técnica antes mencionada (web scraping) para luego ser procesada y/o transformada a través de scripts. Todo lo anterior es realizados en el lenguaje de programación Python. Posteriormente, la información resultante puede ser enviada hacia bases de datos y/o archivos tales como hojas de cálculo u otras páginas web para ser consultada y tratada de acuerdo con las necesidades de los analistas de reportes.

En este informe también se menciona un plan de contingencia por parte de la empresa, producto de un incidente de ciberseguridad contra sus plataformas e infraestructura informática. Cabe mencionar que lo anterior debe ser tomado como parte del aprendizaje profesional y se hace evidente que este tipo de eventualidades no son ajenas en el entorno profesional y empresarial actual, por lo tanto, se hace importante contar con planes y acciones que permitan proteger la infraestructura, plataformas, procesos y servicios que conforman el negocio de la empresa.

1 Planteamiento del problema

Emtelco es una empresa que ofrece servicios de BPO. Esto se puede interpretar como: externalización de procesos del negocio, en consecuencia, la empresa se enfoca en brindar apoyo a determinados procesos informáticos y digitales para otras empresas interesadas. Como parte de estos servicios se presenta el área de programación y control dentro del cual, parte de su trabajo es generar reportes a partir de información existente en sitios web cuyo contenido es generalmente producto de encuestas, estadísticas, inquietudes, quejas, solicitudes, etc. de clientes de las empresas que usan los servicios de Emtelco.

Teniendo en cuenta lo anteriormente mencionado, se hacen más evidentes las preguntas: ¿Cómo se generan estos reportes? ¿Existe suficiente capacidad laboral humana capaz de procesar la información requerida por los negocios y empresas interesados?

Es de estas preguntas que nace la necesidad de realizar tareas de manera automática, en este caso, generar reportes a partir de la información proveniente de sitios web y que estos logren ser de utilidad para los analistas correspondientes.

La automatización de reportes a partir de datos contenidos en sitios web, posee como base el uso de algunas librerías o algoritmos que permitan realizar técnicas para la extracción de información sin necesidad de hacerlo manualmente. Una de estas técnicas es el web scraping y es la que se ha empleado durante la realización de las labores de la práctica profesional.

1.1 Antecedentes

Desde la masificación del internet y aún más, con los continuos avances de las técnicas y tecnologías que conforman el mundo digital, se hace cada vez más complejo o incluso inviable, la extracción manual de información que se encuentra alojada en sitios web. Es por esta razón que paralelamente al crecimiento de la cantidad de información existente en internet, se desarrollan métodos para la obtención de datos e información de la manera más automática posible. En la actualidad, uno de los conceptos en los que mayormente se trabaja e investiga para desarrollar y aplicar en el campo de la extracción, análisis y automatización de grandes cantidades de datos es la inteligencia artificial, lo cual está haciendo que las técnicas de web scraping sean reemplazadas

o fusionadas con técnicas y aplicaciones basadas en inteligencias artificiales y probablemente en un corto plazo este sea el medio de automatización más efectivo.

2 Justificación

La finalidad principal en este proyecto obedece a registrar el trabajo realizado durante el periodo de la práctica académica profesional. Dicha práctica, la cual se realizó mediante la modalidad de semestre de industria, fue realizada y concluida de manera exitosa en la empresa Emtelco, realizando labores de automatización de reportes. La finalidad era obtener información proveniente de sitios web y llevarla a los formatos requeridos después de su tratamiento adecuado para que finalmente esta fuera apta para el uso de los analistas de reportes correspondientes.

3 Objetivos

3.1 Objetivo general

Realizar actividades de automatización en extracción de datos de sitios web y su posterior filtrado y transformación en información relevante mediante el uso de tecnologías tales como Selenium y técnicas como Web scraping, así como la formulación y creación de instrucciones y algoritmos usando el lenguaje Python para acceder y extraer el contenido de dichas páginas, transformándola luego en información útil para los clientes, de manera automática.

3.2 Objetivos específicos

- Identificar los elementos de las páginas web que deben ser manipulados a través del lenguaje de programación Python.
- Implementar la librería Selenium a través del lenguaje Python para que se permita la automatización de los elementos de las páginas web correspondientes.
- Elegir la lógica de programación adecuada en el lenguaje Python para el tratamiento, filtrado, transformación y salida de reportes de las páginas web que vayan a ser sometidas a procesos de acceso con automatización.
- Implementar tecnologías como el componente Selenium WebDriver y técnicas como Web scraping para acceder y extraer contenido de las páginas web.
- Desarrollar código y algoritmos usando el lenguaje Python para integrar la lógica y las librerías requeridas en los procesos de automatización.

4 Problema de investigación

Utilizar e implementar las herramientas adecuadas para la automatización de reportes de la manera requerida por el área de programación y control de la compañía Emtelco. Estas herramientas eran principalmente las contenidas en las librerías del componente Selenium WebDriver, las cuales son herramientas para desarrollar actividades de web scraping. También es apropiado mencionar que no solo estas herramientas formaron parte del conjunto de elementos que debieron ser aprendidos para realizar un trabajo exitoso, sino que también, durante el proceso de la práctica, se trabajó con otros tipos de tecnologías en las cuales se adquirió conocimiento y experiencia como fueron plataformas para el desarrollo en el lenguaje Python. Dichas plataformas eran *Jupyter* y *Spyder*, los cuales son entornos para programación. También se requirió de tiempo para comprender los procesos de solicitud, creación y entrega de automatizaciones y esto se hizo posible con el consejo e inducción de los profesionales a cargo.

5 Marco teórico

La automatización de recolección de datos de páginas web, se ha ido convirtiendo en una necesidad debido a la gran cantidad de información a ser procesada. Esto ha llevado a desarrollar técnicas para dichos procedimientos. Uno de los focos principales de esta propuesta se encuentra en una técnica denominada como “web scraping”.

Esta técnica de automatización permite ingresar, manipular y extraer información de sitios web por medio de algoritmos, programas o librerías que simulan la navegación humana a través de las páginas de los mencionados sitios.

Web scraping es el proceso de recolectar datos contenidos en páginas web mediante técnicas automatizadas. Lo distintivo del web scraping es que en principio los datos parecen poco estructurados. Corresponde por tanto al analista de datos identificar cuál es el patrón que siguen los datos, para luego crear y ejecutar un algoritmo de extracción y procesamiento de estos. En la práctica lo que se hace es escribir un programa que envía consultas a un servidor web, recibe las respuestas (usualmente en forma de páginas web) y examina los datos para extraer la información necesaria (Mitchell, 2015).

Durante la práctica empresarial, aprender a utilizar y poner en ejecución dicha técnica de automatización tiene como objetivo, permitir acceder a sitios web autorizados por los clientes de la empresa, para la extracción de datos. Los cuales podrán ser procesados posteriormente y así generar reportes con información relevante. Todo este proceso se realizará desde el lenguaje de programación Python y utilizando la librería “Selenium WebDriver”, componente del entorno de pruebas para aplicaciones basadas en web Selenium, el cual contiene dentro de su ecosistema las herramientas apropiadas para la automatización en el acceso y manipulación dentro de sitios web y así mismo todo lo necesario para ser integrada dentro del lenguaje Python.

6 Metodología

La metodología para alcanzar los objetivos planteados consiste en implementar adecuadamente las funciones contenidas dentro de la librería Selenium y así mismo realizar una lógica correcta en la programación de algoritmos dentro del lenguaje Python para el filtrado y transformación de los datos. Posteriormente, se realizarán pruebas para verificar el correcto funcionamiento de la automatización y finalmente se pondrá en ejecución dentro de un contexto de producción real. A continuación, se listan las fases más importantes que conforman la ejecución de la metodología:

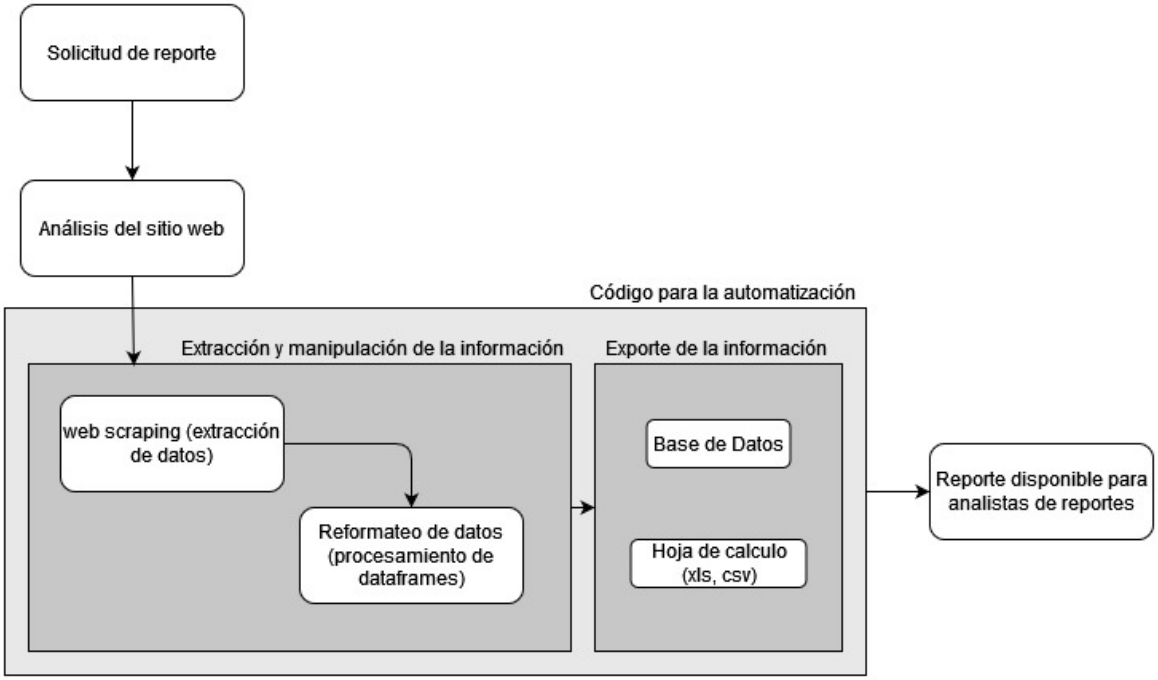
1. Recibir solicitudes de automatización a través de tickets. Esto es un sistema implementado por la empresa para agilizar el inicio de nuevos proyectos solicitados.
2. Revisar, analizar y comprender la solicitud de automatización incluida en el ticket.

   Ejemplo: Se recibe una solicitud de automatización para extraer los reportes que resultan de seleccionar determinadas opciones dentro de un formulario en un sitio web. En este caso se debe estudiar y decidir cómo se implementará el código para que, al ser ejecutado, se lance el sitio web deseado y se seleccionen las opciones indicadas dentro del ticket, se descargue el reporte para su posterior procesamiento con el lenguaje Python y se retorne un reporte con las transformaciones requeridas.
3. Desarrollar e implementar el código para la automatización En el lenguaje Python, incluyendo tanto la lógica para acceder al sitio web y extraer la información, como la lógica para manipular y transformar los datos extraídos.
4. Realizar las pruebas correspondientes para verificar el correcto funcionamiento de todo el modelo de la automatización.
5. Informar dentro del tiempo establecido sobre la culminación y correcto funcionamiento del código de automatización y aguardar al visto bueno del encargado de autorizar la puesta en producción del código.
6. Incluir el código en las tareas programadas del sistema operativo para que este sea ejecutado automáticamente a las horas predefinidas.

En la **Figura 1.** Se puede visualizar el flujo del desarrollo de un proyecto de automatización.

Figura 1.

Diagrama de flujo de un proyecto de automatización de reporte



7 Resultados

Se logró adquirir conocimientos y experiencia en procesos de automatización de sistemas informáticos, más específicamente, automatización en la extracción de datos de sitios web, filtrado y transformación de conjuntos de datos. Así mismo se han ampliado los conocimientos en el uso de las herramientas y técnicas usadas para los mencionados procesos para finalmente integrar lo aprendido en un futuro en el área laboral.

Adicionalmente, se ha adquirido una experiencia en el ámbito profesional referente a los percances y situaciones inesperadas, tal como lo fue el ciber ataque a las plataformas informáticas de la empresa, llevando esto a tomar medidas de manera ágil y formar parte en la planificación de contingencias para disminuir y eliminar el daño ocasionado, lo cual se toma como una experiencia valiosa para futuros contextos que se asemejen.

La práctica académica inició con una inducción y entrenamiento durante las primeras cinco semanas, la cual es impartida por uno de los ingenieros del área de programación y control. Durante dicho periodo se reforzó y extendió el conocimiento en el lenguaje Python, en especial en el uso de la librería Pandas y de tablas mediante la función dataframe. También se realizó una introducción al componente *Selenium WebDriver* perteneciente al entorno de pruebas Selenium, el cual se implementa dentro del ambiente de Python igualmente.

Ya pasadas las semanas de inducción y entrenamiento, se inicia con la participación en un primer proyecto, el cual ya estaba existente, pero requería intervención para ser puesto en funcionamiento debido a que en su estado actual presentaba problemas en su código y en las funciones destinadas para acceder y controlar el sitio web, por tal motivo, este proyecto fue un muy buen punto de acercamiento a las labores que se seguirían realizando en la empresa. Dicho proyecto está denominado como *“timeshift”*, el cual es también el nombre del sitio web desde el cual se debe extraer la información para generar posteriormente el reporte.

El código inicia con el importe de las librerías apropiadas para el funcionamiento del código, como se puede observar en la Figura 2.

Figura 2.

```python
import os
import sys
import time
import shutil
from datetime import date
from openpyxl import *
from pyautogui import *
import datetime
import pandas as pd
import numpy as np
import locale
from selenium import webdriver
from pyautogui import press, sleep
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver import ActionChains
chrome_options = webdriver.ChromeOptions()
# chrome_options.add_argument("--headless") # Runs Chrome in headless mode.
chrome_options.add_argument('--no-sandbox')  # Bypass OS security model
chrome_options.add_argument('--disable-gpu')  # applicable to windows os only
chrome_options.add_argument('start-maximized')  #
chrome_options.add_argument('disable-infobars')
chrome_options.add_argument("--disable-extensions")
WINDOW_SIZE = "3000,3000"
chrome_options.add_experimental_option("prefs", {
    "download.default_directory": r"G:\Descargas\Canales virtuales\02. Tymeshift\Descargas",
    "download.prompt_for_download": False,
    "download.directory_upgrade": True,
    "safebrowsing.enabled": True
})
chrome_options.add_argument("--window-size=%s" % WINDOW_SIZE)

browser = webdriver.Chrome(ChromeDriverManager().install(), options=chrome_options)

locale.setlocale(locale.LC_ALL, 'es-ES')

n = 1 #DIAS
# m = 0
# for n in range (n,m,-1):


today = datetime.datetime.now()
```

A continuación, como se puede observar en la Figura 3, se usan funciones para preparar el inicio del navegador web al momento de ejecutar el código y se preparan variables para controlar fechas que serán necesarias al momento de filtrar datos en el sitio web. Así mismo se prepara un directorio en una ubicación de servidor determinada, en la cual el conjunto de reportes será guardado permanentemente, siendo este nombrado con la fecha en la que se crearán los reportes.

Figura 3.

```
today = datetime.datetime.now()
today=today-datetime.timedelta(days=n)
today_mes=today.strftime('%B')
today_mes=today_mes.capitalize()
sleep(1)
os.makedirs('//10.1.1.7/01 Oficina Planeación y Control/01 Analytics/Mabe/Zendesk/Time/'+today_mes,exist_ok=True)
sleep(1)
os.makedirs('//10.1.1.7/01 Oficina Planeación y Control/01 Analytics/Mabe/Zendesk/Time/'+today_mes+'/'+today.strftime('%d'),


def codigo(browser):
    #frm = ['CO :: B2C :: WhatsApp móvil',
     #       'CO :: Retención Móvil', 'CO :: WhatsApp new stack', 'CO :: eCare new stack',
      #      'Co :: B2c :: Soporte Hogar', 'Cambio de domicilio',
       #     'Eje Cafetero', 'Retención Hogar', 'Whatsapp Hogares', 'Soporte Técnico',
        #    'Co :: B2b :: Soporte', 'Co :: B2c :: Soporte Móvil', 'Pequeñas Empresas', 'CO :: B2C :: VENTAS',
         #   'CO :: B2B :: VENTAS', 'CO :: B2B :: RETENCIÓN', 'Co :: B2b :: Grandes Empresas', 'Co :: B2b :: Ecav']

    frm = ['CO :: B2C :: WhatsApp móvil', 'CO :: Retención Móvil',
           'Co :: B2c :: Soporte Hogar', 'Cambio de domicilio',
           'Eje Cafetero', 'Retención Hogar', 'Whatsapp Hogares', 'Soporte Técnico',
           'Co :: B2b :: Soporte', 'Co :: B2c :: Soporte Móvil', 'Pequeñas Empresas', 'CO :: B2C :: VENTAS',
           'CO :: B2B :: VENTAS', 'CO :: B2B :: RETENCIÓN', 'Co :: B2b :: Grandes Empresas', 'Co :: B2b :: Ecav']


    browser.get('https://us2.tymeapp.com/v2/login?user=tigopy&domain=tigopy.tymeapp.com')
    browser.maximize_window()
    sleep(5)
    email = WebDriverWait(browser, 10).until(EC.element_to_be_clickable((By.XPATH, '//*[@id="email"]/div/div/input')))
    #//*[@id="email"]/div/div/input
    #//*[@id="email"]/div/div/div/input
    # email.send_keys([redacted])
    email.send_keys([redacted])
    password = WebDriverWait(browser, 10).until(EC.element_to_be_clickable((By.XPATH, '//*[@id="password"]/div/div/input')))
    #//*[@id="password"]/div/div/div/input
    #//*[@id="password"]/div/div/input
    # password.send_keys([redacted])
    password.send_keys([redacted])
    WebDriverWait(browser, 10).until(
        EC.element_to_be_clickable((By.XPATH, '//*[@id="app"]/div/div[2]/form/div[3]/div/div/div/div/button/span'))).click()
    print("entro")
    WebDriverWait(browser, 10).until(EC.element_to_be_clickable(
        (By.XPATH, '//*[@id="main-container"]/div[1]/div[1]/div/div/div/ul[1]/li[4]/a'))).click()
    WebDriverWait(browser, 10).until(
        EC.element_to_be_clickable((By.XPATH, '//*[@id="content"]/div[2]/div[2]/div/div[1]/div'))).click()
    print("paso1")
    # -------- METRICAS ----------
    sleep(3)
```

Seguidamente, se tiene una función llamada *"codigo"*. Esta tiene un parámetro llamado *"browser"* el cual llama el controlador necesario para iniciar el navegador web Chrome y sus correspondientes opciones iniciales. Dentro de la función se crea una lista que corresponde a un menú determinado del sitio web, que será usado posteriormente para descargar cada uno de los reportes que corresponden a cada nombre de la lista, también se tiene la dirección web que se desea abrir, se envían las credenciales para el inicio de sesión y demás comandos para continuar con la simulación de estar navegando dentro del sitio web (Figura 3).

En la siguiente parte del script, continuando dentro de la función *"codigo"*, se recorre la lista anteriormente mencionada para así, ir filtrando, seleccionando y descargando los reportes del día seleccionado (Figura 4).

Figura 4.

```
    WebDriverWait(browser, 10).until(
        EC.element_to_be_clickable((By.XPATH, '//*[@id="parameters"]/div/div/div[1]/div/div[1]'))).click()
    sleep(1)
    a = 1
    while a == 1:
        print('entró al while')
        try:
            sleep(1)
            WebDriverWait(browser, 10).until(EC.element_to_be_clickable(
                (By.XPATH, '//*[@id="main-container"]/div[1]/div[1]/div/div/div/ul[2]/li[1]/a'))).click()
            sleep(1)
            WebDriverWait(browser, 10).until(EC.element_to_be_clickable((By.XPATH,'//*[@id="main-container"]/div[1]/div[1]/c
            sleep(5)
        except:
            a = 2
    carpeta = r"G:\Descargas\Canales virtuales\02. Tymeshift\Descargas"
    os.chdir(carpeta)
    for item in os.listdir(carpeta):
        nombrearchivo = os.path.abspath(item)
        os.remove(nombrearchivo)

    print("entro al sumary")
    for i in frm:
        sleep(2)
        WebDriverWait(browser, 10).until(
            EC.element_to_be_clickable((By.XPATH, '//*[@id="s2id_group_row13"]/a/span[2]'))).click()
        sleep(2)
        buscador = WebDriverWait(browser, 10).until(
            EC.element_to_be_clickable((By.CSS_SELECTOR, '.select2-input.select2-focused')))
        sleep(1)
        buscador.send_keys(i)
        sleep(2)
        browser.find_elements_by_class_name('select2-result-label')[0].click()
        csv = WebDriverWait(browser, 10).until(
            EC.element_to_be_clickable((By.XPATH, '//*[@id="print_area"]/div[1]/div/div[5]/button')))
        csv.click()
        sleep(3)
        csv = WebDriverWait(browser, 10).until(
            EC.element_to_be_clickable((By.XPATH, '//*[@id="print_area"]/div[1]/div/div[5]/ul/li[1]/a')))
        sleep(3)
        csv.click()
        print("Descargando")
        carpeta = r"G:\Descargas\Canales virtuales\02. Tymeshift\Descargas"
        os.chdir(carpeta)
        extension = ".csv"
        while not os.listdir(carpeta) != []:
```

También se controlan ciertas condiciones para que el script no se vaya a detener en algún punto de la ejecución (Figura 5). En este punto cabe señalar que los reportes serán inicialmente descargados en una ubicación del equipo local, es decir, en el computador que está ejecutando el código, para posteriormente ser enviados a la ubicación del servidor.

Figura 5.

```
        while not os.listdir(carpeta) != []:
            try:
                print("entro a la carpeta")
                sleep(15)  # //*[@id="main-container"]/div[1]/div[1]/div/div/div/ul[2]/li[1]/a
                WebDriverWait(browser, 10).until(EC.element_to_be_clickable(
                    (By.XPATH, '//*[@id="main-container"]/div[1]/div[1]/div/div/div/ul[2]/li[1]/a'))).click()
                sleep(3)
                WebDriverWait(browser, 10).until(EC.element_to_be_clickable((By.XPATH,
                                                                             '//*[@id="main-container"]/div[1]/div[1]/div/div
                sleep(10)
            except:
                continue
        while os.listdir(carpeta)[0].endswith(extension) is False:
            sleep(4)
        carpeta = r"G:\Descargas\Canales virtuales\02. Tymeshift\Descargas"
        nombre_file = i

        if i == 'CO :: B2C :: WhatsApp móvil':
            ruta = '//10.1.1.7/01 Oficina Planeación y Control/01 Analytics/Mabe/Zendesk/Time/'+today_mes+'/'+today.strftime
            nombre_file = nombre_file.replace('CO :: B2C :: WhatsApp móvil', "WhatsApp móvil")



        elif i == 'CO :: Retención Móvil':
            ruta = '//10.1.1.7/01 Oficina Planeación y Control/01 Analytics/Mabe/Zendesk/Time/'+today_mes+'/'+today.strftime
            nombre_file = nombre_file.replace('CO :: Retención Móvil', "Retención Móvil")

        #elif i == 'CO :: WhatsApp new stack':
         #   ruta = '//10.1.1.7/01 Oficina Planeación y Control/01 Analytics/Mabe/Zendesk/Time/'+today_mes+'/'+today.strftim
          #  nombre_file = nombre_file.replace('CO :: WhatsApp new stack', "WhatsApp new stack")

        #if i == 'CO :: eCare new stack':
         #   ruta = '//10.1.1.7/01 Oficina Planeación y Control/01 Analytics/Mabe/Zendesk/Time/'+today_mes+'/'+today.strftim
          #  nombre_file = nombre_file.replace('CO :: eCare new stack', "eCare new stack")



        elif i == 'Co :: B2c :: Soporte Hogar':
            ruta = '//10.1.1.7/01 Oficina Planeación y Control/01 Analytics/Mabe/Zendesk/Time/'+today_mes+'/'+today.strftime
            nombre_file = nombre_file.replace('Co :: B2c :: Soporte Hogar', "Soporte Hogar")

        elif i == 'Co :: B2b :: Soporte':
            ruta = '//10.1.1.7/01 Oficina Planeación y Control/01 Analytics/Mabe/Zendesk/Time/'+today_mes+'/'+today.strftime
            nombre_file = nombre_file.replace('Co :: B2b :: Soporte', "Soporte")

        elif i == 'Co :: B2c :: Soporte Móvil':
            ruta = '//10.1.1.7/01 Oficina Planeación y Control/01 Analytics/Mabe/Zendesk/Time/'+today_mes+'/'+today.strftime
            nombre_file = nombre_file.replace('Co :: B2c :: Soporte Móvil', "Soporte Movil")
```

El segundo proyecto denominado como *"E-care"*, similar al anterior, aunque algo más corto en contenido de código, es una automatización ya existente, la cual presentaba inconvenientes de acceso al sitio web, problemas con los formatos de fechas y errores para la descarga de los reportes. Las correcciones, al igual que en el primer proyecto, fueron exitosas.

El script inicia con el importe de las librerías necesarias. Posteriormente se continúa con una función llamada *"save"*, dentro de la cual se preparan variables para las fechas, carpetas locales y del servidor. También se tiene un ciclo para verificar los reportes existentes en la carpeta local y moverlos al servidor (Figura 6).

Figura 6.

```python
# -*- coding: utf-8 -*-

import os
import sys
import ast
import time
import glob
import shutil
import datetime
import numpy as np
import pandas as pd
from selenium import webdriver
from pyautogui import press, sleep
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager



def save(a):
    today = datetime.date.today()
    ayer = today - datetime.timedelta(days=1)
    ruta = '//10.1.1.7/01 Oficina Planeación y Control/01 Analytics/Mabe/Medallia/' + str(a)
    carpeta = r'G:\Descargas\Medalia\02. Ecare\Descargas'
    ext = ".xlsx"
    os.chdir(carpeta) #cambia el directorio de trabajo a la carpeta

    while not os.listdir(carpeta)!=[]:
        sleep(2)
    while os.listdir(carpeta)[0].endswith(ext) is False:
        sleep(3)


    for item in os.listdir(carpeta): #recorrer todos los items de la carpeta
            if item.endswith(ext): #comprueba la extensión .zip
                nombrearchivo= os.path.abspath(item)
                shutil.copy(nombrearchivo,ruta)
                os.remove(nombrearchivo)

```

A continuación, se define la función *"options"*, la cual contiene todos los pasos para filtrar y seleccionar los datos solicitados para conformar el reporte. Dentro de esta función se llama a la función anterior (*"save"*) para el almacenamiento persistente del reporte (Figura 7). Seguidamente, se preparan las instrucciones para inicializar el navegador web a través de las funciones de *Selenium WebDriver* al igual que en el proyecto anterior (Figura 8). Finalmente se llama a la función *"options"* para que ejecute la automatización (Figura 9).

Figura7.

```
def options(i):
    sleep(15)
    WebDriverWait(browser, 2).until(EC.element_to_be_clickable((By.XPATH, '//*[@id="ReactRoot"]/div[2]/div/main/div[1]/div/d
    sleep(5)
    item = WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,'//*[@id="timeperiod-control"]')))
    item.click()
    sleep(10)
    '''
    try:
        WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.CSS_SELECTOR,'//div[text()="Cancelar"]'))).click()
    except:
        print("no encontro el boton cancelar")
    finally:
    '''


    #---Codigo para descargar por Dia
    sleep(2)
    WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,
                                                                 '//*[@id="ReactRoot"]/div[2]/div/main/div[1]/div/form/div/d
    hoy = datetime.datetime.now()
    antier = hoy - datetime.timedelta(days=1)
    antier2 = hoy - datetime.timedelta(days=1)
    f_antier = antier.strftime('%d/%m/%y')
    f_antier2 = antier2.strftime('%d/%m/%y')

    inicio = WebDriverWait(browser, 20).until(
        EC.presence_of_all_elements_located((By.CSS_SELECTOR, '[placeholder="dd/mm/yy"]')))[0]
    fin = WebDriverWait(browser, 20).until(
        EC.presence_of_all_elements_located((By.CSS_SELECTOR, '[placeholder="dd/mm/yy"]')))[1]
    inicio.send_keys(f_antier)
    fin.send_keys(f_antier2)
    #---Termina Aca---#
    """
    #sleep(10)
    ayer =  WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,'//button[text()="Ayer"]')))
    ayer.click()
    """
    sleep(10)
    Run = WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,'//*[@id="ReactRoot"]/div[2]/div/main/div[1]
    Run.click()
    sleep(4)
    Des = WebDriverWait(browser, 2).until(EC.element_to_be_clickable((By.CSS_SELECTOR, '[aria-label="Exportar"]')))
    Des.click()
    sleep(4)
    Exc = browser.find_element_by_xpath('//button[text()="Excel"]')
    Exc.click()
    save(i)
```

Figura 8.

```
chrome_options = webdriver.ChromeOptions()
#chrome_options.add_argument("--headless") # Runs Chrome in headless mode.
chrome_options.add_argument('--no-sandbox') # Bypass OS security model
chrome_options.add_argument('--disable-gpu')  # applicable to windows os only
chrome_options.add_argument('start-maximized') #
chrome_options.add_argument('disable-infobars')
chrome_options.add_argument("--disable-extensions")
WINDOW_SIZE = "3000,3000"

chrome_options.add_experimental_option("prefs", {
"download.default_directory": r"G:\Descargas\Medalia\02. Ecare\Descargas",
"download.prompt_for_download": False,
"download.directory_upgrade": True,
"safebrowsing.enabled": True
})

chrome_options.add_argument("--window-size=%s" % WINDOW_SIZE)


browser = webdriver.Chrome(ChromeDriverManager().install(), options = chrome_options)

carpeta = r"G:\Descargas\Medalia\02. Ecare\Descargas"
os.chdir(carpeta)
for item in os.listdir(carpeta):
    nombrearchivo= os.path.abspath(item)
    os.remove(nombrearchivo)

browser.get('https://tigo.medallia.com/tigo/pages/412?roleId=32356&f.question-score=k_tigo_digital_ltr&f.timeperiod=318&f.re
browser.maximize_window()
## Ingresar Usuario y Contraseña
username = browser.find_element_by_name([redacted])
#username.send_keys([redacted])#[redacted]
#username.send_keys([redacted])
username.send_keys([redacted])
sleep(1)
password = browser.find_element_by_name('password')
#password.send_keys([redacted]#user1
#password.send_keys([redacted])
password.send_keys([redacted])
login = browser.find_element_by_id('logBtn')
login.click()
sleep(4)

file = open(r"G:\Descargas\Medalia\02. Ecare\Ecare.txt", "r")
contents = file.read()
dictionary = ast.literal_eval(contents)
file.close()
```

Figura 9.

```
for i in dictionary:
    try:
        browser.get(dictionary[i])
        options(i)
        sleep(1)
    except Exception as e:
        print(e)
        browser.get(dictionary[i])
        options(i)
        sleep(1)
        continue

browser.quit()
```

El tercer proyecto, *"rescate virtual"*, es una automatización que se realizó desde cero. El objetivo en este proyecto fue el de realizar un script de automatización de principio a fin y este proceso al igual que los anteriores, fue satisfactorio. Aunque la estructura y el propósito de este proyecto son similares a los anteriores, el hecho de iniciar uno nuevo desde cero, permitió adquirir y expandir la experiencia en los procesos de automatización.

Al igual que en los códigos anteriores, en necesario iniciar con el importe de las librerías requeridas (Figura 10). En este caso, dentro de estos importes, se llama un script denominado *"cargue"*, el cual llama funciones para trabajar con una base de datos, es decir, en este caso no solo se extrae la información a ficheros de tipo hoja de cálculo o de valores separados por coma, sino que también, se almacena en tablas en una base de datos, por lo tanto se realizaron scripts adicionales en ficheros independientes para las conexiones y las funciones respectivas y así tener un mejor entendimiento del código del proyecto. Después del importe de librerías, se crean las variables para establecer las rutas de almacenamiento y las fechas con sus formatos respectivos (Figura 10). Debido a que este proyecto se inició desde cero, se optó por hacer el código más claro por lo cual se decidió hacer la mayoría del script separado por funciones, así como poner en ficheros independientes, la conexión hacia la base de datos. A continuación, se describen estas funciones y demás instrucciones que conforman el resto del código.

Figura 10.

```python
from pyautogui import *
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from webdriver_manager.chrome import ChromeDriverManager
import pandas as pd
import shutil
from datetime import date, time, datetime
from openpyxl import Workbook
from cargue import *
import datetime
import numpy as np
from os import remove
from automagica import empty_folder
```

```python
empty_folder(r'G:\Descargas\RescateV\descargas')
empty_folder(r'G:\Descargas\RescateV\temporal')
ruta=r"G:\Descargas\RescateV\descargas"
carpeta="G:\Descargas\RescateV\temporal"
hoy = datetime.date.today()
ayer = hoy - datetime.timedelta(days=1)
#ayer=ayer.strftime('%Y-%m-%d')
#ayer2=ayer.strftime('%d/%m/%Y')
#ayer1=ayer.strftime('%Y/%m/%d')
#ayer2=ayer.strftime('16/01/2022')
#ayer1=ayer.strftime('09/01/2022')
ayer1=ayer.strftime('2022/03/09')
ayer2=ayer.strftime('2022/03/16')
flag=False
```

Las funciones *"fechaInicialInput"* y *"fechaFinalInput"* tienen como propósito insertar una fecha en el formato adecuado, en los campos correspondientes del sitio web para filtrar los datos requeridos en un rango de fechas (Figura 11).

Figura 11.

```
#Poner fecha inicial en campo adecuado
def fechaInicialInput(f1):
    estado ="error"
    while estado == "error":
        try:
            #Limpiar campo de fecha inicial
            WebDriverWait(browser, 5).until(
                EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl04_ctl03_txtValue"]'))).clear()
            #Llenar campo de fecha inicial con fecha del dia anterior
            WebDriverWait(browser, 5).until(
                EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl04_ctl03_txtValue"]'))).send_keys(f1)
            #Enter para confirmar fecha
            #WebDriverWait(browser, 5).until(
             #   EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl04_ctl03_txtValue"]'))).send_keys(Keys.ENTER)
            estado="hecho"
        except Exception as e:
            print("Fecha: Esto puede tardar, espere por favor...")
            #  print ("Opción invalida")
            #print (e)
            estado="error"

#Poner fecha final en campo adecuado
def fechaFinalInput(f1):
    estado ="error"
    while estado == "error":
        try:
            #Limpiar campo de fecha final
            WebDriverWait(browser, 5).until(
                EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl04_ctl05_txtValue"]'))).click()
            WebDriverWait(browser, 5).until(
                EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl04_ctl05_txtValue"]'))).clear()
            #Llenar campo de fecha final
            WebDriverWait(browser, 5).until(
                EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl04_ctl05_txtValue"]'))).send_keys(f1 + r' 23:59:59')
            #Enter para confirmar fecha
            #WebDriverWait(browser, 5).until(
             #   EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl04_ctl05_txtValue"]'))).send_keys(Keys.ENTER)
            estado="hecho"
        except Exception as e:
            print("Fecha2: Esto puede tardar, espere por favor...")
            #  print ("Opción invalida")
            #print (e)
            estado="error"

```

La función *"programaSelect"* realiza las acciones de desplegar una lista y seleccionar la opción adecuada en el espacio del sitio web (Figura 12).

Seguidamente, se define la función *"verInformeButton"*, la cual solo tiene el objetivo de activar las acciones de un botón del sitio web para mostrar los resultados de la selección anterior (Figura 12).

Se continúa con la función *"menuExportarButton"*. Con esta función se despliega un menú que contiene las opciones para descargar el reporte en varios formatos y se selecciona descargar en formato de hoja de cálculo de extensión xlsx (Figura 12).

Figura 12.

```
#Seleccionar opcion adecuada en campo "PROGRAMA"
def programaSelect():
    estado= "error"
    while estado== "error":
        try:
            #Click en menu "consultar por"
            WebDriverWait(browser, 5).until(
                EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl04_ctl07_ddValue"]'))).click()
            #Seleccionar opción del menú
            WebDriverWait(browser, 5).until(
                EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl04_ctl07_ddValue"]'))).send_keys(Keys.ARROW_DOWN,
                                                                                                         Keys.ENTER)
            estado= "hecho"
        except Exception as e:
            print("Programa: Esto puede tardar, espere por favor...")
            estado="error"

#Hacer click en boton "Ver informe"
def verInformeButton():
    estado= "error"
    while estado== "error":
        try:
            #Click en botón "ver informe"
            WebDriverWait(browser, 5).until(
                EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl04_ctl00"]'))).click()
            estado= "hecho"
        except Exception as e:
            print("Boton ver: Esto puede tardar, espere por favor...")
            estado="error"

#Acciones para exportar archivo
def menuExportarButton():
    estado= "error"
    while estado== "error":
        try:
            #Click en menú "exportar"
            WebDriverWait(browser, 5).until(
                EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl05_ctl04_ctl00_ButtonLink"]'))).click()
            sleep(2)
            #Clickc en opcion "exportar archivo Excel"
            WebDriverWait(browser, 5).until(
                EC.element_to_be_clickable((By.XPATH,'//*[@id="ctl32_ctl05_ctl04_ctl00_Menu"]/div[5]/a'))).click()
            estado= "hecho"
            print("Menu exportar: Inicio de la descarga")
        except Exception as e:
            print("Menu exportar: Esto puede tardar, espere por favor...")
            estado="error"

```

Para las siguientes tres funciones, se debe dejar claro que, estas realizan una acción similar, con la diferencia que cada una de ellas se ejecuta para distintas páginas de un sitio web que contienen tres diferentes reportes, es decir, los datos son diferentes pero su estructura es similar. Las funciones en cuestión son denominadas: *"filtrarDatosTipificacion"*, *"filtrarDatosDetallado"* y *"filtrarDatosIndicadores"* (Figura 13, Figura 14 y Figura 15). Estas funciones cumplen el papel de cargar los archivos correspondientes a un *dataFrame* para ser modificados de acuerdo con lo requerido. Las funciones retornan un dataframe modificado para ser cargado posteriormente a la base de datos.

Figura 13.

```python
def filtrarDatosDetallado():
    estado= "error"
    while estado== "error":
        try:
            ###Cargar archivo a data frame###
            df= pd.read_excel(fr"G:\Descargas\RescateV\temporal\01_Reporte_Detallado_Rescate_Virtual.xlsx", skiprows=15) # o
            cols= ['Program', 'Call Id', 'Fecha Inicial', 'Fecha Final',
                   'Identificador Agente Chat', 'Key', 'Identificador Chat',
                  'Fecha Evento', 'Origen Evento', 'Usuario Evento', 'Dialogo Chat']
            df_detallado=df[cols]
            df_detallado['Program']= df_detallado['Program'].fillna(method='ffill')
            df_detallado= df_detallado.drop(0)

            df_detallado.columns=(df_detallado.columns).str.lower()
            df_detallado.columns = df_detallado.columns.str.replace(" ","_")
            df_detallado.columns = df_detallado.columns.str.replace("identificador","id")
            df_detallado["dialogo_chat"]=df_detallado["dialogo_chat"][0:3000]
            ###Descarga de archivos###
            df_detallado.to_excel(fr"{ruta}\Detallado_rescate_virtual_chat.xlsx", index=False)
            estado= "hecho"
            print("Se procesó y descargó exitosamente detallado!")
            flag=True

        except Exception as e:
            print("Cargando y procesando archivo. Espere por favor...")
            sleep(20)
            estado="error"
            flag=False
    return df_detallado
```

Figura 14.

```python
def filtrarDatosTipificacion():
    estado= "error"
    while estado== "error":
        try:
            ###Cargar archivo a data frame###
            df= pd.read_excel(fr"G:\Descargas\RescateV\temporal\02_Reporte_Tipificacion_Rescate_Virtual.xlsx", skiprows=15)
            cols= ['Call Id', 'Program', 'Identificador Chat', 'Documento',
                   'Usuario', 'Nombre Usuario', 'Duración', 'Fecha', 'Hora',
                   'Nivel1', 'Nivel2', 'Nivel3', 'Calificación Encuesta']
            df_tipificacion=df[cols]
            df_tipificacion.columns=(df_tipificacion.columns).str.lower()
            df_tipificacion.columns = df_tipificacion.columns.str.replace(" ","_")
            df_tipificacion.columns = df_tipificacion.columns.str.replace("identificador","id")
            df_tipificacion.columns = df_tipificacion.columns.str.replace("ó","o")
            ###Descarga de archivos###
            df_tipificacion.to_excel(fr"{ruta}\Tipificacion_rescate_virtual.xlsx", index=False)
            estado= "hecho"
            print("Se procesó y descargó exitosamente tipificacion!")

        except Exception as e:
            print("Cargando y procesando archivo. Espere por favor...")
            sleep(20)
            estado="error"
    return df_tipificacion
```

Figura 15.

```python
def filtrarDatosIndicadores():
    estado= "error"
    while estado== "error":
        try:
            ###Cargar archivo a data frame###
            df= pd.read_excel(fr"G:\Descargas\RescateV\temporal\03_Reporte_Indicadores_Rescate_Virtual.xlsx", skiprows=15) #
            cols= ['Fecha', 'Program', 'Hora', 'Interacciones entrantes',
                   'Interacciones contestadas', 'Interacciones abandonadas',
                   'Porcentaje abandono', 'Porcentaje eficacia',
                   'Nivel servicio', 'Tiempo promedio conversando',
                   'Tiempo promedio ASA', 'Tiempo promedio Primera Respuesta',
                   'Promedio agentes', 'Tiempo total conversando',
                   'Total Interacciones']
            df_indicadores=df[cols]
            df_indicadores['Fecha']= df_indicadores['Fecha'].fillna(method='ffill')
            df_indicadores.columns=(df_indicadores.columns).str.lower()
            df_indicadores.columns = df_indicadores.columns.str.replace(" ","_")
            df_indicadores.columns = df_indicadores.columns.str.replace("identificador","id")
            df_indicadores.columns = df_indicadores.columns.str.replace("ó","o")
            df_indicadores.rename(columns={'tiempo_promedio_conversando':'tiempo_conversando' }, inplace=True)
            df_indicadores.rename(columns={'tiempo_promedio_asa':'tiempo_asa' }, inplace=True)
            df_indicadores.rename(columns={'tiempo_promedio_primera_respuesta':'tiempo_primera_respuesta' }, inplace=True)
            ###Descarga de archivos###
            df_indicadores.to_excel(fr"{ruta}\Indicadores_rescate_virtual.xlsx", index=False)
            estado= "hecho"
            print("Se procesó y descargó exitosamente Indicadores!")
        except Exception as e:
            print("Cargando y procesando archivo. Espere por favor...")
            sleep(25)
            estado="error"
    return df_indicadores
```

Después de haber definido las funciones anteriores, se inicia con las variables y comandos para preparar el inicio de los sitios web (Figura 16 y Figura 17). Debido a que este script va a hacer uso de tres diferentes direcciones web, se requiere que cada vez que se inicie un sitio y cumpla su función, este se cierre para continuar con el siguiente. Así entonces, se inicia un sitio web, se llama a las funciones requeridas (*"fechaInicialInput"*, *"fechaFinalInput"*, *"programaSelect"*, *"verInformeButton"*, *"menuExportarButton"* y las funciones para filtrar los datos correspondientes para cada dirección) y se cierra el navegador web.

Figura 16.

```
chrome_options = webdriver.ChromeOptions()

prefs= {"download.default_directory" : r"G:\Descargas\RescateV\temporal",
        "directory_upgrade" : True,
        "download.prompt_for_download": False,
        "download.directory_upgrade": True,
        "safebrowsing.enabled": True
       }
chrome_options.add_experimental_option("prefs", prefs)

#chrome_options.add_argument("--headless")  # Runs Chrome in headless mode.
chrome_options.add_argument('--no-sandbox')  # Bypass OS security model
chrome_options.add_argument('--disable-gpu')  # applicable to windows os only
chrome_options.add_argument('start-maximized')  #
chrome_options.add_argument('disable-infobars')
chrome_options.add_argument("--disable-extensions")
WINDOW_SIZE = "3000,3000"
chrome_options.add_argument("--window-size=%s" % WINDOW_SIZE)
browser = webdriver.Chrome(ChromeDriverManager().install(), options= chrome_options)  # Inicializar Chrome
#browser = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=chrome_options)
browser.get('http://reportesemtelco/reportes/Pages/Report.aspx?ItemPath=%2fUNE%2fAplicaciones%2fRescate+Virtual%2f01_Reporte
browser.maximize_window()

###Inicio funciones archivo Detallado_rescate_virtual.xlsx###
fechaInicialInput(ayer1)
fechaFinalInput(ayer2)
programaSelect()
verInformeButton()
menuExportarButton()
df_det=filtrarDatosDetallado()
browser.quit()
###fin funciones archivo Detallado_rescate_virtual.xlsx###
#browser = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=chrome_options)
browser = webdriver.Chrome(ChromeDriverManager().install(), options= chrome_options)  # Inicializar Chrome
browser.get('http://reportesemtelco/reportes/Pages/Report.aspx?ItemPath=%2fUNE%2fAplicaciones%2fRescate+Virtual%2f02_Reporte
browser.maximize_window()
```

Figura 17.

```
###Inicio funciones archivo Tipificacion_rescate_virtual.xlsx###
fechaInicialInput(ayer1)
fechaFinalInput(ayer2)
programaSelect()
verInformeButton()
menuExportarButton()
sleep(10)
df_tip=filtrarDatosTipificacion()
browser.quit()

###Fin funciones archivo Tipificacion_rescate_virtual.xlsx###
browser = webdriver.Chrome(ChromeDriverManager().install(), options= chrome_options)  # Inicializar Chrome
browser.get('http://reportesemtelco/reportes/Pages/Report.aspx?ItemPath=%2fUNE%2fAplicaciones%2fRescate+Virtual%2f03_Reporte
browser.maximize_window()
fechaInicialInput(ayer1)
fechaFinalInput(ayer2)
programaSelect()
verInformeButton()
sleep(5)
menuExportarButton()
df_indi=filtrarDatosIndicadores()
browser.quit()
```

```
#df_det=filtrarDatosDetallado()
cargue_detallado(ayer1,ayer2,df_det)
```

```
cargue_tipificacion(ayer1,ayer2,df_tip)
```

```
cargue_indicadores(ayer1,ayer2,df_indi)
```

Para la última parte del script se hace un llamado a las funciones que llevan los datos de los dataframe a la base de datos (Figura 17). Estas funciones se encuentran en el fichero *"cargue"* (Figura 18) que fue importado al inicio del código principal. Cabe anotar que dentro del código de "cargue" se llama a su vez a otro fichero que contiene la conexión a la base de datos (Figura 19).

Figura 18.

```
from Loadtopsql2 import *
import pandas as pd


def cargue_detallado(df):
    delete = f"DELETE FROM public.rescate_virtual_detallado WHERE date(fecha_inicial) BETWEEN'{primerdia}'and '{dia_pasado}'
    #delete = f"DELETE FROM public.confiar_campa_unicos WHERE fecha_contacto BETWEEN'{primerdia}'and '{dia_pasado}'and campa

    insert = """insert into public.rescate_virtual_detallado(program, call_id, fecha_inicial, fecha_final, id_agente_chat,
                                                              key, id_chat, fecha_evento, origen_evento, usuario_evento,
                                                              dialogo_chat, created_by)
                                                      VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"""

    return load(df, delete, insert) + ' para la tabla public.interaxa_colas_intervalo.\n'


# ## LLAMADAS ENTRANTES

def cargue_tipificacion(primerdia,dia_pasado, df):
    delete = f"DELETE FROM public.rescate_virtual_indic WHERE date(fechahoraingreso) BETWEEN'{primerdia}'and '{dia_pasado}'"
    #delete = f"DELETE FROM public.confiar_campa_llamadas WHERE fecha_contacto BETWEEN'{primerdia}'and '{dia_pasado}'and cam

    insert = """insert into public.rescate_virtual_indic(call_id, program, id_chat, documento, usuario,
                                                          nombre_usuario, duracion, fecha, hora, nivel1, nivel2,
                                                          nivel3, calificacion_encuesta, created_by)
                                                  VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"""

    return load(df, delete, insert) + ' para la tabla public.rescate_virtual_indic.\n'

def cargue_indicadores(primerdia,dia_pasado, df):
    delete = f"DELETE FROM public.rescate_virtual_indic WHERE date(fechahoraingreso) BETWEEN'{primerdia}'and '{dia_pasado}'"
    #delete = f"DELETE FROM public.confiar_campa_llamadas WHERE fecha_contacto BETWEEN'{primerdia}'and '{dia_pasado}'and cam

    insert = """insert into public.rescate_virtual_indic(fecha, program, hora, interacciones_entrantes,
                                                          interacciones_contestadas, interacciones_abandonadas,
                                                          porcentaje_abandono, porcentaje_eficacia, nivel_servicio,
                                                          tiempo_promedio_conversando, tiempo_promedio_asa,
                                                          tiempo_promedio_primera_respuesta, promedio_agentes,
                                                          tiempo_total_conversando, total_interacciones, created_by)
                                                  VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"""

    return load(df, delete, insert) + ' para la tabla rescate_virtual_indic.interaxa_encuestas.\n'
```

Figura 19.

```
# -*- coding: utf-8 -*-

'''
 * Loadtopsql2
 * Carga información al servidor de posgre 10.1.1.25
 *
 * @author scorpere
 *
'''

import pandas as pd
import numpy as np
import psycopg2
import sys
import getpass


def load(df, delete, insert):
    rep = ''
    try:
        connection = psycopg2.connect(user=[redacted]
                                      password=[redacted],
                                      host=[redacted],
                                      port=[redacted],
                                      database=[redacted])
        cursor = connection.cursor()

        df['By'] = getpass.getuser()

        cursor.execute(delete)

        record_to_insert = df.values
        cursor.executemany(insert, record_to_insert)
        rep = str(cursor.rowcount) + " Record inserted successfully"
        connection.commit()
        connection.close()
        cursor.close()

    except Exception as e:
        print('end')
        connection.rollback()
        connection.close()
        cursor.close()
        print(e)
        rep = ('fail : ' + str(e))

    finally:
        return rep
```

El cuarto y último proyecto es denominado *"e-survey"* y es la segunda automatización que se realizó desde cero, aunque para algunas partes de este se usó como ayuda un proyecto anterior similar.

Se importan las librerías y junto a estas, un fichero denominado *"descarga_esurvey"* (Figura 20) para hacer uso de su función *"eformat"*, la cual contiene la mayor parte de la funcionalidad del código. Se define una función llamada *"main"* dentro de la cual se encuentra todo lo necesario para la ejecución de la automatización (Figura 20, Figura 21 y Figura 22). Se preparan: el acceso a los directorios en los que se descarga la información del sitio web, variables para enviar un correo electrónico al momento de ejecución del script y así validar su funcionamiento, un usuario y contraseña extraído de un fichero externo para validar el ingreso al sitio y algunas opciones para la inicialización del navegador web. Posteriormente se llama a la función *"eformat"* (Figura 22), perteneciente al fichero *"descarga_esurvey"*, la cual contiene, entre otra lógica, todos los comandos de *Selenium WebDriver* para simular la navegación por el sitio web y así seleccionar y descargar la información requerida. Dicha función tiene cinco parámetros: una variable con las opciones para inicializar el navegador web, usuario, contraseña, correo electrónico y mensaje del correo electrónico. El usuario y la contraseña son las credenciales para iniciar sesión en el sitio web del cual se extraen los datos para el reporte.

Figura 20.

```
'''
 ' Descarga archivos para informe migraciones.
 '
 ' @author EduardoB
 '
'''
import import_ipynb
import Descarga_Esurvey_Exito as descarga_esurvey
import os
import shutil
import pandas as pd
from cargue import *
#from Descarga_Exito_Esurvey import eformat
from pyautogui import  sleep
from selenium import webdriver
from automagica import empty_folder
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.support import expected_conditions as EC


message1 = """\
From: No Reply <no_reply@automatizacion.com>
To: Person <no_reply@automatizacion.com>
Subject: Descarga Migraciones

No se logra procesar la descarga de migraciones.

Plataforma:
"""

def main():
    try:
        os.chdir(r'D:\pruebas')
        #os.chdir(r'D:\Reportes\Migraciones\00. Main\Solo Esurvey')
        os.makedirs(os.getcwd()+'\\Descargas',exist_ok=True)
        os.makedirs(os.getcwd()+'\\Temp',exist_ok=True)
```

Figura 21.

```
        chrome_options = webdriver.ChromeOptions()
        #chrome_options.add_argument("--headless") # Runs Chrome in headless mode.
        chrome_options.add_argument('--no-sandbox') # Bypass OS security model
        chrome_options.add_argument('--disable-gpu')  # applicable to windows os only
        chrome_options.add_argument('start-maximized') #
        chrome_options.add_argument('disable-infobars')
        chrome_options.add_argument("--disable-extensions")
        chrome_options.add_argument("--lang=es-US")
        WINDOW_SIZE = "3000,3000"

        chrome_options.add_experimental_option("prefs", {
        "download.default_directory": os.getcwd()+'\\Temp',
        "download.prompt_for_download": False,
        "download.directory_upgrade": True,
        "safebrowsing.enabled": True
        })

        chrome_options.add_argument("--window-size=%s" % WINDOW_SIZE)


        browser = webdriver.Chrome(ChromeDriverManager(
                        ).install(),options=chrome_options)

        #

        empty_folder(os.getcwd()+'\\Temp')
        empty_folder(os.getcwd()+'\\Descargas')


        #df = pd.read_excel(os.getcwd()+'\\Esurvey.xlsx')
        df = pd.read_excel(os.getcwd()+'\\Esurvey_exito.xlsx')
        #df.index = df['Aplicativo']
        df.index = df['aplicativo']
        myemail = df.dropna(subset=['correos'])
        myemail = myemail['correos'].values
        #myemail = [
        print(myemail)
```

Figura 22.

```
        descarga_esurvey.eformat(browser,df['usuario']['Esurvey'],df['contraseña']['Esurvey'],myemail,message1)

        #ruta = r'\\10.1.1.7\01 Oficina Planeación y Control\32-GTR\45 - MESA DE DESPACHO\Automatizacion Despacho'
        ruta = r'D:\pruebas\rutafinal'

        for item in os.listdir(os.getcwd()+'\\Descargas'): #recorrer todos los items de la carpeta
                #if item.endswith(ext): #comprueba la extensión
                try:
                    name= os.path.join(os.getcwd()+'\\Descargas\\'+item)#obtiene la ruta completa de los archivos
                    shutil.copy(name,ruta)
                except Exception as e:
                    continue

    except Exception as e:
        #browser.close()
        print(e)

    finally:
        browser.quit()
        #contactame1(myemail,message1)
        print("it's ok")

if __name__ == "__main__":
    main()
```

Más detalladamente, la función *"eformat"* (Figura 23), aparte de contener la lógica para la simulación de la navegación en el sitio web, también hace el llamado a dos funciones: *"fechas"* y *"guardar"*. La función *"fechas"* da formato e inserta la fecha para la descarga del reporte en los campos adecuados del sitio web (Figura 24). La función *"guardar"*, extrae la información de los archivos descargados del sitio web en un *dataFrame* para procesarlos y adecuarlos de acuerdo con lo requerido y guardarlos de nuevo ya modificados (Figura 25).

Figura 23.

```
def eformat(browser,usuario,contraseña,myemail,message1):
    try:
        #browser.get('https://esurveydespacho.emtelco.co/home/forms/')
        browser.get('https://esurveyexito.emtelco.co/home/forms/')
        browser.maximize_window()
        user = WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,
                                                '//*[@id="login"]/div[1]/div[2]/div[2]/div[1]/div/div/div/div[2]/div[1]/inpu
        user.send_keys(usuario)
        passw = WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,
                                                '//*[@id="login"]/div[1]/div[2]/div[2]/div[2]/div/div/div/div[2]/div[1]/inpu
        passw.send_keys(contraseña)
        log = WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,
                                                '//*[@id="login"]/div[1]/div[2]/div[2]/button/div[2]/div'))).click()
        sleep(5)
        form = WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,
                                    '//*[@id="q-app"]/div/div[1]/aside/div[2]/a/div/div/div')))
        form.click()
        sleep(1)

        #lista = ['Auto Migraciones - Control Ing','Escalamientos Infraestructura AAA', 'Novedades Premisas.','Nivelacion De
        #lista = ['Novedades Premisas.','Nivelacion De Rutas Premisas','Casos Críticos Dispatcher']
        lista = ['Tipificación Grupo Exito SAC','Tipificación Ventas Viajes']

        for i in lista:
            try:
                filtro=WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,'//*[@id="listform"]/div[2]/div
                filtro.send_keys(i)
                sleep(2)
                WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,'//i[text()="search"]'))).click()
                sleep(3)

                if i == 'Tipificación Grupo Exito SAC' :
                    print("entro al if")
                    cloud = WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,
                                                '//*[@id="listform"]/div[2]/div[3]/div/div/div[1]/div/div/div[4]

                else:
                    cloud = WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,
                                                '//*[@id="listform"]/div[2]/div[3]/div/div/div[1]/div/div/div[4]


                today = datetime.date.today()
                ndias = today-datetime.timedelta(days=16)
                quince = 15
                hoy = today.day
                ndias = ndias.day

                cloud.click()
                fechas(hoy,hoy,True,browser)
                sleep(1)
                guardar(1,i)

                df= pd.read_excel(os.getcwd() +'\\Descargas\\'+i+'_'+str(1)+'.xlsx')
                df.columns=(df.columns).str.lower()
                df.columns = df.columns.str.replace(" ","_")
                df.to_excel(os.getcwd() +fr'\\Descargas\\'+i+'_1'+'.xlsx', index=False)
                #print(df)

                filtro.clear()

            except Exception as e:
                print(e)
                sleep(10)
                filtro.clear()
                continue

        cargue_tipificacion_sac(hoy, hoy, df)
    except Exception as e:
        info2 = myemail
        message1 = message1 + 'Esurvey' + '\n' +str(e)
        smtpObj = smtplib.SMTP([redacted])
        smtpObj.sendmail(sender, info2, message1)
        print("Successfully sent email")
        print('Error...Esurvey')
        print(e)
```

Figura 24.

```
def fechas(x,y,z,browser):

    inicial=WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,'/html/body/div[3]/div[2]/div[3]/div/div/d
    sleep(1)

    if z == True:
        #izquierda=browser.find_element_by_xpath('/html/body/div[4]/div/div[2]/div/div[2]/div[1]/button[1]/div[2]/i')
        #izquierda.click()#click en la flecha para ir al mes anterior
        sleep(2)

    try:
        WebDriverWait(browser, 20).until(EC.presence_of_all_elements_located((By.XPATH, '//span[text()="'+str(y)+'"]')))[1].
    except:
        WebDriverWait(browser, 20).until(EC.presence_of_all_elements_located((By.XPATH, '//span[text()="'+str(y)+'"]')))[0].
    sleep(2)
    final=WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH,'/html/body/div[3]/div[2]/div[3]/div/div/div
    sleep(2)
```

Figura 25.

```
def guardar(var,name):
    print(str(var),name)
    carpeta = os.getcwd()+'\\Temp'
    ext = ".xlsx"

    if name == 'Auto Migraciones - Control Ing':
        ruta= os.getcwd() +'\\Descargas\\MigracionesEsurvey_'+str(var)+'.xlsx'
        print("Aqui entra a guardar archivos exito1")
    else:
        ruta= os.getcwd() +'\\Descargas\\'+name+'_'+str(var)+'.xlsx'
        print("Aqui entra a guardar archivos exito2")

    while not os.listdir(carpeta)!=[]:
        sleep(2)
    while os.listdir(carpeta)[0].endswith(ext) is False:
        sleep(2)

    for item in os.listdir(carpeta): #recorrer todos los items de la carpeta
        if item.endswith(ext): #comprueba la extensión
            nombre= os.path.join(os.getcwd()+'\\Temp\\'+item)#obtiene la ruta completa de los archivos
            shutil.copy(nombre,ruta)
        empty_folder(carpeta)

    if var == 2 and name != 'Auto Migraciones - Control Ing' :
        print('ingreso concat2')

        df = pd.read_excel(os.getcwd() +'\\Descargas\\'+name+'_1.xlsx',dtype={'Id llamada 1':str,'Id llamada 2':str})
        dff = pd.read_excel(os.getcwd() +'\\Descargas\\'+name+'_2.xlsx',dtype={'Id llamada 1':str,'Id llamada 2':str})
        df2 = pd.concat([df,dff],ignore_index=True)
        df2.to_excel(os.getcwd() +'\\Descargas\\'+name+'_1.xlsx',index = False)
        os.remove(os.getcwd() +'\\Descargas\\'+name+'_2.xlsx')

        print('save')

    elif name == 'Auto Migraciones - Control Ing' and var == 2:
        print('ingreso concat')
        df = pd.read_excel(os.getcwd() +'\\Descargas\\MigracionesEsurvey_1.xlsx',dtype={'Id llamada 1':str,'Id llamada 2':st
        dff = pd.read_excel(os.getcwd() +'\\Descargas\\MigracionesEsurvey_2.xlsx',dtype={'Id llamada 1':str,'Id llamada 2':s
        df2 = pd.concat([df,dff],ignore_index=True)
        print(df2.info())
        #df2.to_excel(os.getcwd() +'\\Descargas\\MigracionesEsurvey_1.xlsx',index = False)

        print('save')
        #os.remove(os.getcwd() +'\\Descargas\\MigracionesEsurvey_2.xlsx')
        print('end')

    else:
        print('cero')

    print("Esurvey it's ok")
```

Adicionalmente dentro de la función *"eformat"*, se carga un dataframe con los archivos que contienen los reportes para realizar una modificación de reemplazo de espacios en blanco y luego llevar esta información a una base de datos a través de la función de cargue, similar al script del tercer proyecto (*"rescate virtual"*). La función de carga a la base de datos se define en un fichero independiente (Figura 26).

Figura 26.

```python
#!/usr/bin/env python
# coding: utf-8

from Loadtopsql2 import *
import pandas as pd


def cargue_tipificacion_sac(primerdia,dia_pasado, df):
    df = df.fillna(np.nan).replace([np.nan], [None])
    delete = f"DELETE FROM nombre_de_tabla WHERE date(fecha_inicial) BETWEEN'{primerdia}'and '{dia_pasado}'"
    #delete = f"DELETE FROM public.confiar_campa_unicos WHERE fecha_contacto BETWEEN'{primerdia}'and '{dia_pasado}'and campania = '13908'"

    insert = """insert into nombre_de_tabla(caso, estado, fecha_de_creación, creado_por, fecha_de_modificación, modificado_por,
informacion_del_consumidor, tipo_documento, documento_de_identidad, nombres, apellidos, telefono_fijo, telefono_celular,
cuenta_con_correo_electronico, corre_electronico, departamento, ciudad, id_llamada, informacion_motivo_contacto, canal_de_atencion,
grupo_de_atencion, nivel_1_carulla, nivel_2_carulla, nivel_3_carulla, nivel_1_exito, nivel_2__exito, nivel_3_exito, nivel_1_viva_online,
nivel_2_viva_online, nivel_3_viva_online, nivel_1_domicilios, nivel_2_domicilios, nivel_3_domicilios, nivel_1_novapp, nivel_2_novapp,
nivel_3_novapp, nivel_1_arkitect_y_bronzini, nivel_2_arkitect_y_bronzini, nivel_3_arkitect_y_bronzini, nivel_1_surtimax_super_inter,
nivel_2_surtimax_super_inter, nivel_3_surtimax_super_inter, nivel_1_conmutador, nivel_2_conmutador, nivel_3_conmutador,
nivel_1_llamada_con_novedad, nivel_2_llamada_presenta_novedad, nivel_3_tipo_novedad, nivel_1_llamadas_de_salida, nivel_2_tipo_llamada_salida,
nivel_3_motivo_llamada_salida, otros_negocios, nivel_1_transportadora, nivel_2_transportadora, nivel_3_transportadora, fecha_y_hora_novedad,
telefono_donde_se_comunica_el_cliente, transportadora_aliada, se_creo_caso_en_crm, numero_caso_crm, datos_prioritarios_crm, datos_financieros,
informacion_campo_crm, tipo_gestion_realizada, categoria, numero_de_guia, numero_pedido, cliente_comparte_soporte, observacion,
estado_de_la_novedad, gestion_1, id_llamada_1, gestion_2, id_llamada_2, gestion_3, id_llamada_3, observacion_solucion_final,
numero_caso_crm_trans)
                                                  VALUES
(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,
%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,)"""
    return load(df, delete, insert) + ' para la tabla nombre_de_tabla.\n'

def cargue_venta_viaje(primerdia,dia_pasado, df):
    df = df.fillna(np.nan).replace([np.nan], [None])
    delete = f"DELETE FROM public.rescate_virtual_tipi WHERE date(fecha) BETWEEN'{primerdia}'and '{dia_pasado}'"
    #delete = f"DELETE FROM public.confiar_campa_llamadas WHERE fecha_contacto BETWEEN'{primerdia}'and '{dia_pasado}'and campania = '13908'"

    insert = """insert into public.rescate_virtual_tipi(call_id, program, id_chat, documento, usuario,
                                                  nombre_usuario, duracion, fecha, hora, nivel1, nivel2,
                                                  nivel3, calificacion_encuesta, created_by)
                                               VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"""

    return load(df, delete, insert) + ' para la tabla public.rescate_virtual_tipi.\n'
```

Como conclusión a los resultados, se puede decir que los cuatros proyectos fueron realizados de manera satisfactoria y el conocimiento adquirido se puede considerar que fue creciente en proporción directa con la cronología del tiempo de práctica académica.

11 Recomendaciones

Investigar e implementar técnicas de automatización utilizando herramientas adicionales y paralelas a la librería Selenium, podría tener en consecuencia, una mejora en la eficacia y la eficiencia a la hora de generar reportes, lo que finalmente terminaría beneficiando a cada actor del proceso de los proyectos de automatización de reportes.

Referencias

Mitchell, R., 2015. *Web scraping with Python*. 1st ed. O'Reilly.

Selenium. 2022. *The Selenium Browser Automation Project*. [online] Available at: <https://www.selenium.dev/documentation/>

Anexos

Evidencia de los proyectos realizados durante el periodo de la práctica profesional:

https://drive.google.com/file/d/14WmGZ7d17ikaxCTg_gqdFgqJVksb6Blf/view?usp=sharing