Diseño e implementación de un Bot para validación de landings en el Banco de Bogotá con Python

David Samuel Gutierrez Monroy

Universidad Santo Tomás Seccional Tunja

Facultad de Ingeniería Electrónica (FIE)

Tunja, Colombia

2024

Diseño e implementación de un Bot para validación de landings en el Banco de Bogotá con Python

David Samuel Gutierrez Monroy

Trabajo de grado presentado como requisito para optar al título de:

Ingeniero Electrónico

Director: Ing. Carlos Alberto Cardona Coy

Codirector: Ing. Edgar Andrés Gutierrez Caceres

Universidad Santo Tomás Seccional Tunja

Facultad de Ingeniería Electrónica (FIE)

Tunja, Colombia

2024

Dedicatoria

Agradezco a Dios por ser mi guía y fortaleza diaria, y por darme la salud necesaria para alcanzar mis objetivos en el desarrollo de este proyecto.

A mi familia, por su incondicional apoyo en la realización de este proyecto y por su esfuerzo en hacer realidad la culminación de mi carrera profesional. Su comprensión y ayuda diaria en los diferentes conflictos personales han sido fundamentales, y son mi ejemplo a seguir para mejorar cada día.

Autor

Exoneración de Responsabilidades

Los conceptos desarrollados, las investigaciones realizadas, el análisis y las conclusiones presentadas en este proyecto son responsabilidad exclusiva del autor. Las fuentes originales en las que se fundamenta el documento se citan adecuadamente.

Nota de aceptación:

_____

_____

_____

Firma, jurado 1

_____

Firma, jurado 2

_____

Firma, jurado 3

_____

Tunja, __ de _____, 2024

Agradecimientos

Deseo expresar mi agradecimiento a los ingenieros Cardona y Gutiérrez por su apoyo y guía durante este proyecto y a lo largo de mi desarrollo profesional. Sin su ayuda, la realización de este proyecto no hubiera sido posible.

A la Universidad, por promover mi formación profesional integral y fomentar mi ingenio para la resolución de problemas. Agradezco la exigencia en mi educación y las diversas herramientas brindadas para mi desarrollo profesional.

A mi familia, por su apoyo y motivación incondicionales para la culminación de mi carrera profesional.

Por último, quiero agradecer a mi compañero de carrera, Cristian Cuervo por su apoyo tanto en mi vida académica como personal. Su constante ayuda y motivación fueron fundamentales en momentos de duda y en mi proceso de mejora continua.

Contenido

Figuras

1 Glosario

Software: Hace referencia a programas informáticos los cuales permiten la ejecución de tareas específicas en un sistema de cómputo como lo son aplicaciones, navegadores, u otros. [3]

Robot: Es una pieza de código los cuales cumplen la función ya sea de recopilación de documentos, datos u otros. [10]

Pytest: Pytest es una herramienta de prueba de Python madura y con todas las funciones que le ayuda a escribir mejores programas. [28]

Software de Automatización: Es un software diseñado para convertir tareas repetitivas en acciones automatizadas. [2]

CSV: Es un archivo de texto el cual se caracterizan por sus caracteres separados por comas, haciendo una tabla de filas y columnas. [6]

XPATH: Es una función integrado en selenium el cual permite buscar elementos web los cuales contienen un elemento en particular. [1]

WebDriver: Es una herramienta para pruebas automatizadas de aplicaciones web en diversos navegadores, el cual se puede ejecutar JavaScript o Python. [31]

POO: Se basa en el concepto de crear modelos para la resolución de un problema en un programa, lo cual disminuye los errores y se utiliza para la reutilización de código. [27]

Librerías de Python: Son una colección de funciones y métodos ya definidos lo cual permite aumentar la funcionalidad básica del lenguaje, estar permiten la reutilización de código u acceso a funciones adicionales sin necesidad de realizar una función desde cero. [9]

2 Resumen

En el Banco de Bogotá se identificó la problemática de la validación manual de recursos generados para diversos aliados, lo cual consumía mucho tiempo y esfuerzo. Para abordar esta situación, el proyecto desarrollado en 2024 propone una solución automatizada utilizando diversas herramientas tecnológicas, con el objetivo de reducir el tiempo empleado en estas tareas y eliminar la necesidad de intervención humana.

Este proyecto presenta un resumen del trabajo realizado, destacando las herramientas y técnicas empleadas para solucionar esta problemática mediante un sistema automatizado. La solución se enfoca en la validación y almacenamiento de recursos (enlaces) utilizando el lenguaje de programación Python y el entorno de desarrollo *PyCharm*. Además, se emplearon librerías de código abierto como *WebDriver*, *Pytest* y *CSV*.

El resultado final es un software diseñado por el autor que valida estos recursos de manera local. El sistema comprueba diversos enlaces proporcionados por el Banco de Bogotá, identificando y corrigiendo posibles errores, ya sea en la base de datos de la entidad o en la generación de los recursos.

3 Introducción

La automatización de procesos implica el utilizar sistemas de software con tal de crear instrucciones y procesos repetitivos los cuales reduzcan o reemplacen la interacción humana con los sistemas. Donde las automatizaciones pueden implicar diversas áreas como DevOps, seguridad, pruebas, supervisión entre otros, las cuales tienen grandes ventajas como la eliminación de tareas manuales, aumento de productividad y prioridad en tareas de mayor impacto. [4]

En diversas empresas se implementa la automatización aislada, esto empleado para una sola área con tal de dar solución a una problemática especifica con tal de cumplir un objetivo especifico y que algunas veces es difícil llevar está automatización a diversos ámbitos este mismo desarrollo. [26]

Con respecto a lo anterior se busca solucionar una problemática especifica para el equipo B2B de la empresa teniendo en cuentea la problemática especifica utilizando las herramientas disponibles para el desarrollo del software los cuales facilitan el desarrollo de la automatización y con ello dar la solución a esta problemática.

4 Planteamiento del problema

4.1 Definición del problema

En la ejecución de tareas repetitivas, como la validación de recursos críticos como las landings page o urls generadas por la entidad Banco de Bogotá, es fundamental garantizar su funcionamiento adecuado. El fallo en estas landings no solo representa una pérdida monetaria para la empresa, sino que también genera insatisfacción entre los aliados comerciales, quienes dependen de estas landings para sus propias estrategias de marketing y ventas. Por lo tanto, es imprescindible identificar y solucionar los problemas en las landings antes de su implementación, con el fin de ofrecer resultados satisfactorios tanto a los aliados como a los potenciales clientes.

Actualmente, la validación de las landings se realiza de manera manual, lo que implica un uso ineficiente de recursos y tiempo. La falta de una herramienta automatizada para este propósito dificulta la detección temprana de fallos y retrasa la entrega de landings funcionales y efectivas.

Por ende, la ausencia de una herramienta de validación automatizada para las landings genera diferentes riesgos y la disminución en la calidad del servicio ofrecido a los aliados. Es necesario desarrollar una solución que permita realizar esta validación garantizando la calidad y la confiabilidad de las landings entregadas.

4.2 Formulación de preguntas

A partir de la propuesta de investigación se formulan las siguientes preguntas:

- ¿Cómo puede el desarrollo de una herramienta automatizada de validación de landings impactar positivamente en la eficiencia y la calidad de las operaciones de la entidad?
- ¿Cuáles son los enfoques metodológicos más adecuados para el desarrollo de una herramienta automatizada de validación de landings?
- ¿Cuáles son las mejores prácticas y tecnologías recomendadas en el ámbito de la programación para la implementación de una herramienta automatizada de validación de landings?

4.3 Delimitación del problema

La entidad Banco de Bogotá proporciona una amplia variedad de recursos, incluyendo landings para la solicitud de créditos y tarjetas de crédito, cuya disponibilidad de fondos varía según los convenios establecidos. En este proyecto, nuestro enfoque se centra en analizar las landings específicas, con el fin de identificar y abordar posibles fallos en su funcionamiento.

4.4 Delimitación del problema tiempo

La duración estimada del proyecto es de 2 meses aproximadamente.

4.5 Delimitación del problema lugar

Se llevará a cabo el desarrollo tanto virtual como presencial en las instalaciones de la Universidad Santo Tomás y el Banco de Bogotá. La presentación del proyecto ante la entidad se realizará de forma virtual para verificar el correcto funcionamiento de la herramienta.

4.6 Delimitación del problema desarrollo de pruebas

Las pruebas de funcionamiento se ejecutarán localmente, utilizando como referencia todas las landings con fallos detectados. En caso necesario, la entidad podrá implementar el proceso de validación de recursos de forma automatizada en el futuro

5 Justificación

En la actualidad, los robots de software están ganando cada vez más relevancia. Estos sistemas tienen la capacidad de emular el comportamiento humano y de interactuar con páginas web. Como resultado, los bots pueden asumir tareas que requieran esta capacidad, lo que facilita la estandarización de procesos. Esta automatización conlleva beneficios económicos y de tiempo significativos, al tiempo que reduce errores y mejora la calidad de los resultados. Al liberar a los empleados de tareas repetitivas, los bots permiten que el personal se enfoque en actividades de mayor prioridad y valor agregado. [22] [25]

En la actualidad, los bots se han convertido en una herramienta fundamental para la mejora operativa en numerosas empresas. Su capacidad para automatizar procesos internos no solo conlleva una reducción significativa de costos, sino también una mejora notable en la eficiencia. Además, están disponibles las 24 horas del día, los 7 días de la semana, lo que garantiza una respuesta rápida y constante según las necesidades del usuario. Los bots tienen la capacidad de manejar grandes volúmenes de información y pueden ser diseñados para responder de manera precisa a las solicitudes específicas de los usuarios, lo que les permite cumplir con eficacia su función designada. [18]

En este caso específico para el Banco de Bogotá, resulta imperativo implementar un bot que facilite y reduzca la carga laboral asociada a la validación de los recursos solicitados por la entidad. Esta herramienta no solo minimizará el desgaste del personal, sino que también garantizará la entrega de resultados con mayor precisión y valor para el Banco de Bogotá.

6 Objetivos Generales y específicos

6.1 Objetivo General

Implementar una herramienta capaz de validar las diversas landings, identificar y registrar sus diferentes fallos, y guardar los datos en un archivo CSV para facilitar el análisis y la validación de posibles soluciones a los problemas encontrados en la generación de los recursos de Credi convenio y tarjeta de crédito de la entidad.

6.2 Objetivos específicos

- Validar los errores más comunes que se presentan en las páginas de destino (landings) del sitio web, como errores 404, errores de servidor o contenido no encontrado.
- Crear funciones que permitan la lectura de las landings, la navegación a través de ellas y la captura de los errores que puedan surgir durante la interacción.
- Establecer un proceso robusto para simular la navegación de un usuario real por las landings y capturar los errores de manera eficiente desarrollando funciones que permitan guardar los datos recopilados de las landings en un archivo CSV, tanto para las landings que se cargan correctamente como para aquellas que presentan errores.
- Establecer un formato claro y consistente para el archivo CSV que facilite el análisis posterior de los datos recopilados donde se capturen correctamente los errores de las landings, guardando adecuadamente en el archivo CSV y que la herramienta sea capaz de manejar diferentes escenarios de navegación y errores.

7 Marco teórico

En este apartado se abordarán temas, conceptos y elementos fundamentales en el desarrollo del proyecto, enmarcando las razones por las cuales se les asigna la importancia en el proyecto.

7.1 Lenguaje de programación

Son instrucciones de las cuales se puede tener una interacción entre humanos y computadores por medio de instrucciones escritas en sintaxis que la computadora entienda, el lenguaje de programación permite procesar información de manera rápida y eficiente. Existen diversos lenguajes de programación como Python, JavaScript, C++ entre otros. Su uso principal es para el desarrollo de aplicaciones móviles o de escritorio o desarrollo web donde se plasman las indicaciones y como realizarlas [24] [30].

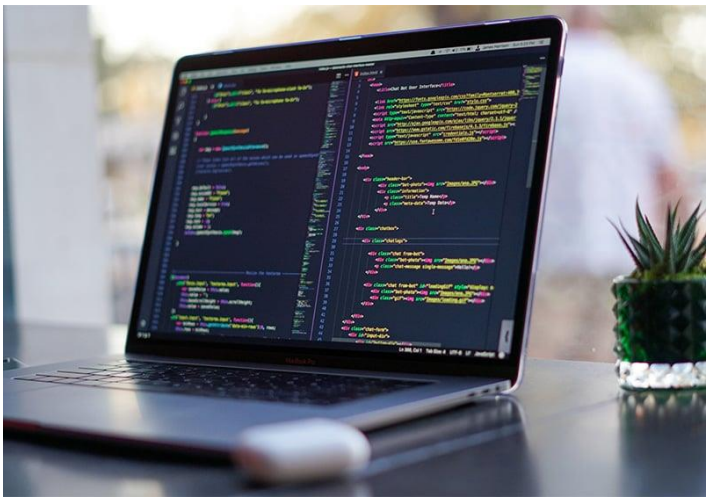


Figura 1. Lenguaje de programación

Nota. *Qué es un lenguaje de programación*. (s.f.). Obtenido de https://openwebinars.net/blog/que-es-un-lenguaje-de-programacion/

7.2 Selenium

Es una herramienta la cual permite ejecutar pruebas automatizadas en aplicaciones web, es una herramienta útil para poder interactuar con elementos de páginas web, y puede ser ejecutada de manera automática e iterativa [19] [33].

Figura 2. Selenium

Nota. *Selenium Python*. (s.f.). Obtenido de https://selenium-python.readthedocs.io/installation.html

7.3 ¿Qué es un bot?

Es una aplicación de software el cual permite automatizar procesos repetitivos de una manera más precisa. Este ejecuta se ejecuta de manera específica para poder imitar el comportamiento humano. [7] [8].

Figura 3. Bots

Nota. ¿Qué es un bot? Tipos de bots y sus principales funciones. (s.f.). Obtenido de https://www.godaddy.com/resources/es/crearweb/que-es-un-bot

7.4 Python

Es un lenguaje de programación utilizado para diversas aplicaciones como webs, software, ciencia de datos entre otros. Este se puede ejecutar en diversas plataformas, es de fácil entendimiento, tiene diversos recursos disponibles para su comprensión, además de su amplia biblioteca el cual contiene diversos códigos los cuales pueden ser reutilizables para ejecutar casi que cualquier tarea. [5] [29].

Figura 4. Python

Nota. Python. (s.f.). Obtenido de https://www.python.org/about/

8 Estado del arte

En la automatización, se destacan diversas categorías de bots, cada una con aplicaciones específicas. Un ejemplo notable es uProc, que ofrece una gama variada de herramientas diseñadas para validar datos como URLs, correos electrónicos, textos, entre otros. Para empresas establecidas, esta diversidad de herramientas representa un beneficio significativo, ya que simplifica y agiliza las diversas verificaciones de datos, optimizando los procesos de validación de información. [20].

Figura 5. Catalogo Web de uProc

Nota. Haz crecer tu negocio con datos optimizados. (s.f.). Obtenido de https://uproc.io/#/?como-tratar-tus-datos-ii-validacion

Además de lo anterior, existen plataformas como ManageEngine que verifican la integridad de los enlaces y proporcionan datos adicionales sobre el sitio web, como su tiempo de actividad, comparativa de enlaces y detección de bloqueos de correos, entre otras funcionalidades. [21]

Figura 6. Web ManageEngine

Nota. Comprobar la existencia de enlaces rotos. (s.f.). Obtenido de https://www.site24x7.com/es/tools/comprobar-de-enlaces.html

9 Diseño metodológico

Este proyecto se realizó por medio de una metodología de aplicación donde se plantea la ruta a seguir del proyecto (se resume en la Figura 7), donde se implementó:

Fase 1: Analizar cuáles son los requisitos para poder realizar el desarrollo del bot.

Fase 2: Adquirir los conocimientos necesarios para poder implementar las herramientas necesarias en el desarrollo del bot.

Fase 3: Analizar las diversas landings para poder mapearlas y saber los errores comunes.

Fase 4: Establecer las herramientas necesarias por medio de la fase 2 y con estos empezar con la implementación en el entorno de programación más favorable.

Fase 5: Verificar las diferentes landings y la llegada hasta estas por medio del bot para la obtención de los errores en estas.

Fase 6: Implementar la obtención del error en cada landing y el guardado en el CSV.

Fase 7: Validar con diversos recursos el guardado del CSV y que corresponda al error de la landing.

Fase 8: Establecer las posibles mejoras.

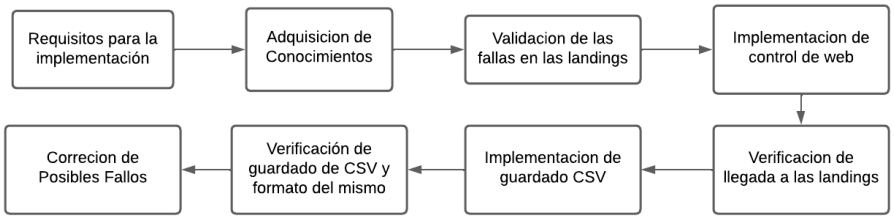


Figura 7. Esquema del diseño metodológico preliminar para el proyecto

Nota. Metodología preliminar. (s.f.). Obtenido de Autor

10 Resultados

El proyecto está conformado únicamente por software, donde principalmente se utilizó la el entorno de programación de PyCharm, donde el lenguaje es Python, adicionalmente del uso de bibliotecas de código abierto de Python. Inicialmente se identificaron las landings de error las cuales eran mas comunes y el flujo el cual se debía seguir para cada uno de los casos tanto en CC (CrediConvenio) como en TC (Tarjeta de Crédito), lo cual estableció la base para poder comenzar con el desarrollo.

10.1 Landings de Error CC

En esta parte inicial se establecieron las landings de error se identificaron que en estas landings se conforman con error "*https://crediconveniodigital.bancodebogota.com.co/error"*, lo cual facilita la verificación de las diferentes landigs debido a que estas están conformadas por la palabra *error*. Por parte del front la respuesta es la de la Fig 8 para poder identificar el tipo de error el cual se va a mapear.

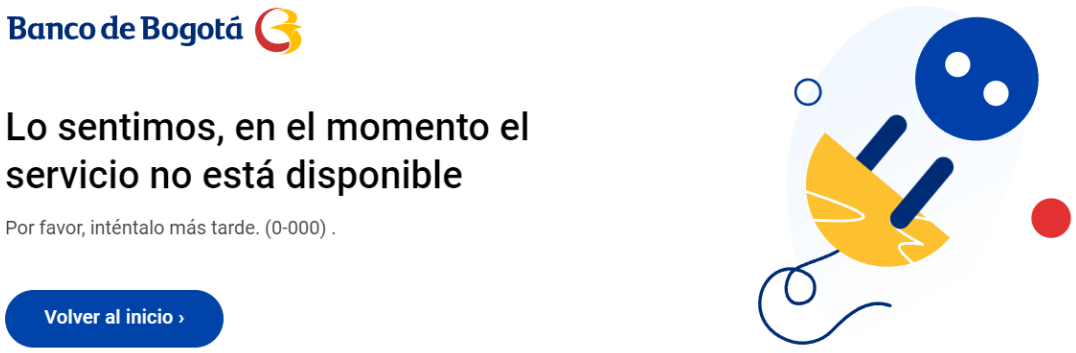


Figura 8. Error Front CC

Nota. Banco de Bogota landing de error CC. (s.f.). Obtenido de https://crediconveniodigital.bancodebogota.com.co/error

10.2 Flujo de Credi-Convenio

En esta parte se identifico el proceso para el flujo tanto normal como de error en CC el cual se evidencia en la Fig 9.

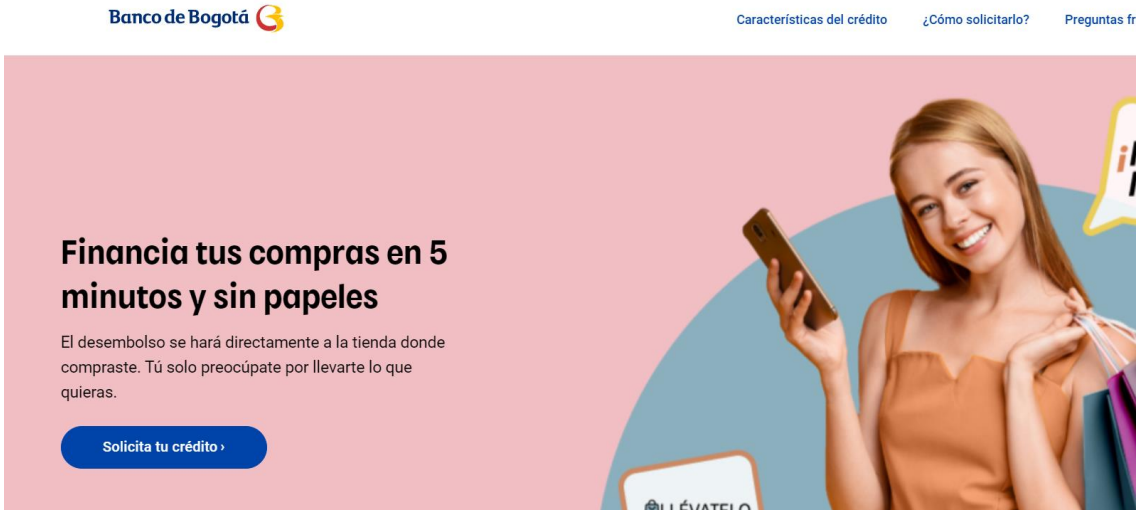


Figura 9. Front CrediConvenio.

Nota. Banco de Bogota landing CC. (s.f.). Obtenido de https://slm.bancodebogota.com/lpu8kpg8

En esta se evidencia que la pagina inicial esta conformada por el botón de solicita tu crédito el cual se daría click y este llevaría a una landing de autenticación y con dos posibles resultados una la cual sigue el flujo normal para la solicitud del crédito como la siguiente "*https://authenticatorweb.bancodebogota.co/?site=”* dando como resultado la Fig 10 y otro con la landing de error "*https://crediconveniodigital.bancodebogota.com.co/error?authLevel=”*.

Figura 10. Front Autenticador para CC

Nota. Banco de Bogota landing CC. (s.f.). Obtenido de https://slm.bancodebogota.com/lpu8kpg8

10.3 Landing de Error TC

En este caso se tienen diferentes flujos para TC por ende se tuvieron en cuenta estos diferentes flujos y errores los cuales se mapearon, donde los mas comunes son la inhabilitación de los links Fig 11 y al igual que en CC la landing de error está conformada por el texto error "*https://tcdigital.bancodebogota.com/error/"* y obteniendo el Front de la Fig 12.

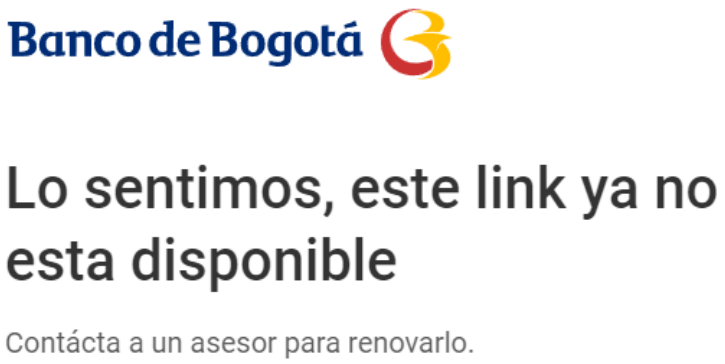


Figura 11. Front Link inhabilitado

Nota. Banco de Bogota landing link no disponible. (s.f.). Obtenido de https://slm.bancodebogota.com/ldewyftn/

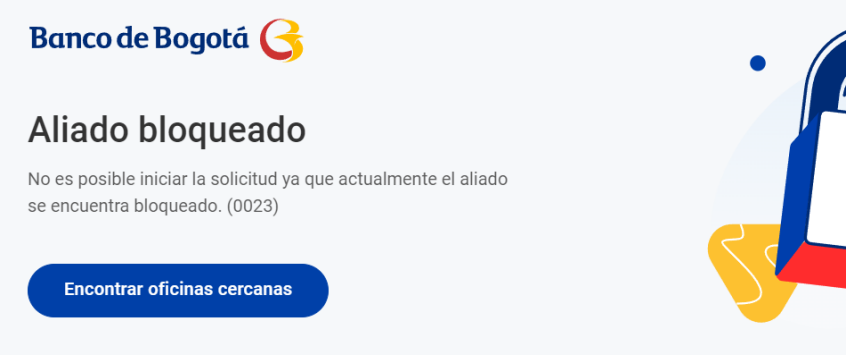


Figura 12. Front de Error

Nota. Banco de Bogota landing TC error. (s.f.). Obtenido de https://tcdigital.bancodebogota.com/error/reject

Anteriormente se evidencian los errores mas comunes en TC y su respectivo Front de su error para poder validar y obtener cual es el error del cual está conformado.

10.4 Flujo de TC

Para este caso se en TC se tienen diferentes webs para la validación del flujo de TC donde se ven en las Fig 13, Fig 14 y Fig 15 respectivamente y se tuvieron en cuenta las diferentes webs para la verificación de estas.

Figura 13. Solicitud TC BDB

Nota. Banco de Bogota landing TC. (s.f.). Obtenido de https://digital.bancodebogota.com/tarjeta-credito/aliados/index.html?utm_source=claro_bajo_costo&utm_medium=referral&utm_campaign=claro_cla_asr57&lowcost=true

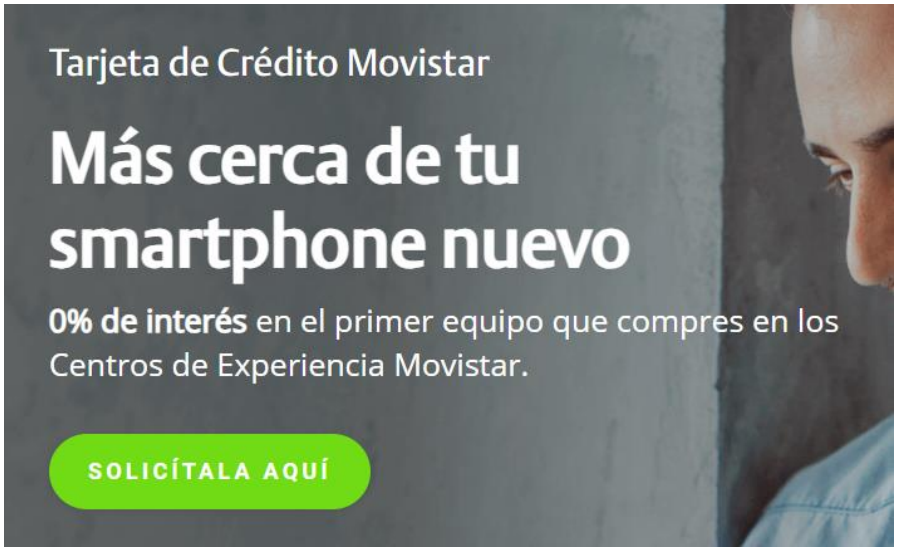


Figura 14. Solicitud TC Movistar

Nota. Banco de Bogota landing TC movistar. (s.f.). Obtenido de https://digital.bancodebogota.com/tarjeta-credito/movistar/index.html

Figura 15. Solicitud TC LATAM

Nota. Banco de Bogota landing TC Latam. (s.f.). Obtenido de https://digital.bancodebogota.co/tarjeta-credito/landing-latampass/index.html

Donde el flujo de TC nos lleva a la solicitud de esta misma como se evidencia en la siguiente imagen.

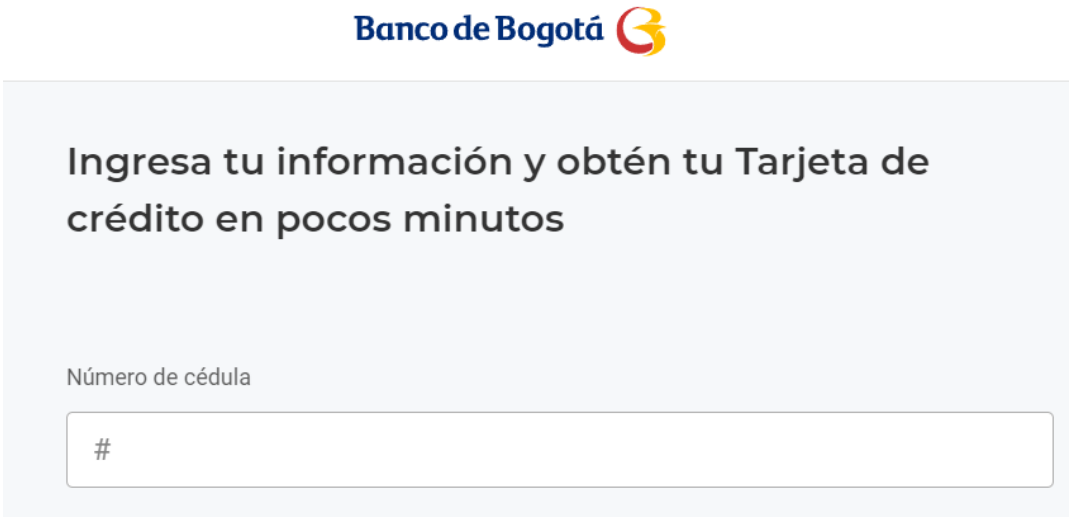


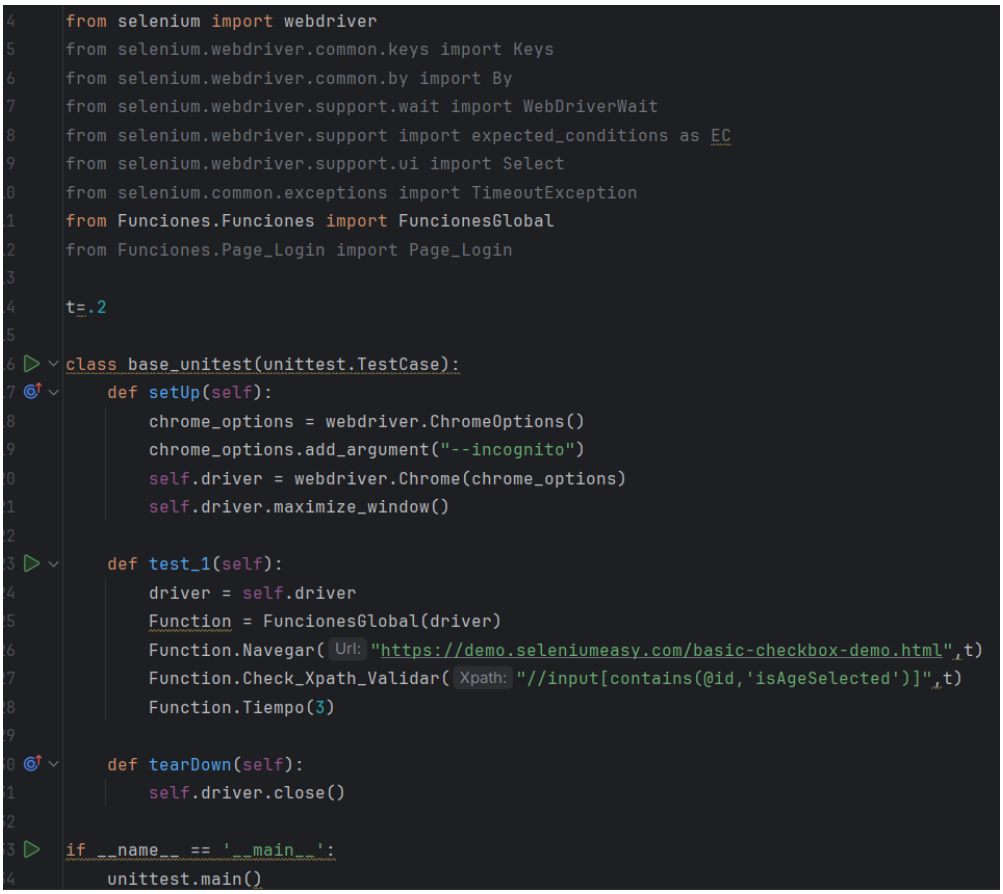
Figura 16. Flujo de solicitud para TC

Nota. Banco de Bogota landing TC. (s.f.). Obtenido de https://digital.bancodebogota.com/tarjeta-credito/aliados/index.html?utm_source=claro_bajo_costo&utm_medium=referral&utm_campaign=claro_cla_asr57&lowcost=true

Teniendo en cuenta las diferentes webs para la solicitud de estas se comenzó a realizar la verificación de estas teniendo en cuenta las diferentes librerías necesarias para el guardado de las landings y control de las diferentes webs.

10.5 Desarrollo de Software

Se comenzó realizando la consulta e implementación de la librería selenium donde se identificaron por medio de este las diferentes acciones necesarias para poder realizar la automatización de las paginas web y obtener los diferentes XPATH de los botones y obtención de las landings en las correspondientes webs para con ello comparar y lograr saber si es satisfactorio o no cada uno de los links. Donde la estructura básica para selenium corresponde a la Fig 17 el cual está conformado por las importaciones de las librerías, opciones del navegador Chrome y el caso a realizar para la validación de las landings.



```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import Select
from selenium.common.exceptions import TimeoutException
from Funciones.Funciones import FuncionesGlobal
from Funciones.Page_Login import Page_Login

t=.2

class base_unitest(unittest.TestCase):
    def setUp(self):
        chrome_options = webdriver.ChromeOptions()
        chrome_options.add_argument("--incognito")
        self.driver = webdriver.Chrome(chrome_options)
        self.driver.maximize_window()

    def test_1(self):
        driver = self.driver
        Function = FuncionesGlobal(driver)
        Function.Navegar( Url: "https://demo.seleniumeasy.com/basic-checkbox-demo.html",t)
        Function.Check_Xpath_Validar( Xpath: "//input[contains(@id,'isAgeSelected')]",t)
        Function.Tiempo(3)

    def tearDown(self):
        self.driver.close()

if __name__ == '__main__':
    unittest.main()
```

Figura 17. Base de Software para selenium

Nota. Base Software. (s.f.). Obtenido de Autor

Para las landing de crediconvenio se comenzó validando el XPATH del botón a seleccionar verificando este y dándole click para lograr comenzar el flujo normal de este siendo *"//button[@data-testid='SolicitaTuCredito'][contains(.,'Solicita tu crédito›')]"*, después de esto se verifica la landing de error *"https://crediconveniodigital.bancodebogota.com.co/error"* y del autenticador *"https://authenticatorweb.bancodebogota.co/"* por medio de funciones, para lograr acortar el código principal y que este sea más limpio aplicando programación orientada a objetos para que este se aplique a cada uno de los links necesarios Fig 18.

```python
def procesar_url(self, url):
    driver = webdriver.Chrome(options=self.chrome_options)
    driver.maximize_window()
    Funcion = FuncionesGlobal(driver)
    try:
        #Funcion.Navegar(url)
        driver.get(url)
        Funcion.Existe( Type: "XPATH", button, Time)
        Funcion.Click_Mixto_Validar( Type: "XPATH", button, Time)
        url_temp = Funcion.ValidarLanding(ErrorURL, urlVerif)
        Funcion.AppendUrl_CC_V2(url_temp, urlVerif, url, NewGoodUrls, NewFailUrls, LandingErrorService,LandingErrorCredito)
    finally:
        driver.quit()
```

Figura 18. Función de validación de landings

Nota. Función landings. (s.f.). Obtenido de Autor

Al verificar que este se esta realizando de manera correcta la validación del link y obtención por medio de consola se procede a realizar el guardado de las diferentes landings con su respectivo error, para este se realiza una función diferente donde por medio de un archivo tipo CSV se extraen los diferentes links y con según el número de landings se cicla en un for para cada uno de los casos y guardando cada uno de ellos en dos archivos correspondiendo en los landing defectuosos y los que si lograron entrar al flujo normal Fig 19.

```python
def test_procesar_urls(self):
    FuncionCSV = FuncionesCSV()
    urls = FuncionCSV.GetCSV(rutaUrls)
    for url_lista in urls:
        for url in url_lista:
            self.procesar_url(url)
```

Figura 19. Función obtención de número de landings

Nota. Función lectura CSV. (s.f.). Obtenido de Autor

Dentro de la verificación de landing en la Fig 18 se tiene la función donde se detecta el error, debido a que el error es un elemento CSS y no se encuentra fácilmente se opto por tomar todo el body correspondiente al error Fig 16 y con ello crear una lista para saber la posición del

error según la landing de error para lograr emplear este en el CSV y con ello saber en que consiste la falla, ya sea por algún error en base de datos u otro como se ve en la Fig 20.

```
try:
    time.sleep(2)
    WebDriverWait(driver, timeout: 10).until(EC.url_contains(LandingError_1))
    elementos_pagina = driver.find_elements(By.CSS_SELECTOR, 'body')
    error = elementos_pagina[0].text
    listerror=error.split()
    Error = listerror[20]
except Exception as e:
    WebDriverWait(driver, timeout: 10).until(EC.url_to_be(LandingError_2))
    elementos_pagina = driver.find_elements(By.CSS_SELECTOR, 'body')
    error = elementos_pagina[0].text
    listerror = error.split()
    Error=listerror[28]
```

Figura 20.Obtención Front

Nota. Obtención front. (s.f.). Obtenido de Autor

Por parte de Tarjeta de crédito se realizó de una manera similar, la diferencia con crediconvenio es la validación en las diferentes webs, por ello se planteó la verificación de estas una por una y con ello usar un solo software para el mismo y no diferirlo en diferentes softwares para cada caso, para este se tomaron los XPATH de los botones en cada uno como lo es para movistar *"/html[1]/body[1]/main[1]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]/button[1]"*, latam *"//button[contains(text(),'ahora ›')]"* y la porpia de Banco de Bogota *"//a[@class='button-request btn-card-request'][contains(.,'Pídela aquí ›')]"*, y después de esto la validación de cada uno como se ve en la Fig 21.

```
def test(self):
    driver=self.driver
    Funcion = FuncionesGlobal(self.driver)
    FuncionCSV = FuncionesCSV()
    urls = FuncionCSV.GetCSV(rutaUrls)
    for landings in urls:
        landing = landings[0]
        Funcion.Navegar(landing)
        Funcion.Click_Xpath_TC(button_moviles, button_movistar, button_others)
        url_temp = Funcion.ValidarLanding(ErrorURL, urlVerif)
        Funcion.AppendUrl_TC_MOV(url_temp, urlVerif, landing, NewGoodUrls, NewFailUrls, LandingErrorBloqueo,LandingErrorCre
```

Figura 21. Función de landing para TC

Nota. Función TC. (s.f.). Obtenido de Autor

Al igual que en crediconvenio se realizó el guardado tanto de la landing como del error al cual pertenece por medio de la misma función y con el mismo principio.

Por último, tanto en TC como en CC se realiza el guardado de las landings con sus errores guardándolo en una carpeta con el mismo formato CSV y comprobando con diferentes links el correcto funcionamiento del software tanto para CC como TC Fig 22, Fig 23 y Fig 24.

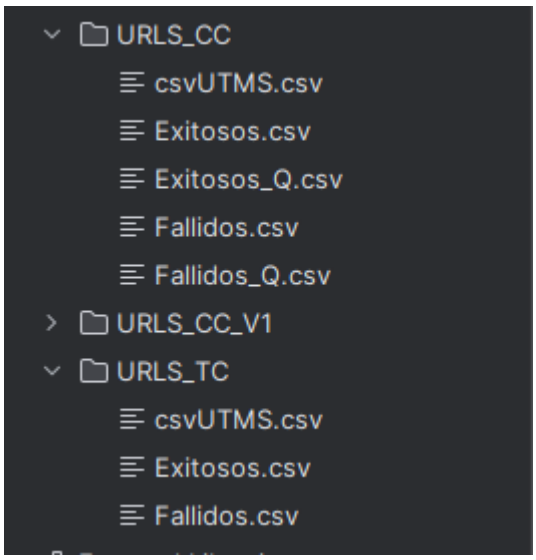


Figura 22. Carpetas CSV exitosos y fallidos

Nota. Ruta guardada CSV. (s.f.). Obtenido de Autor

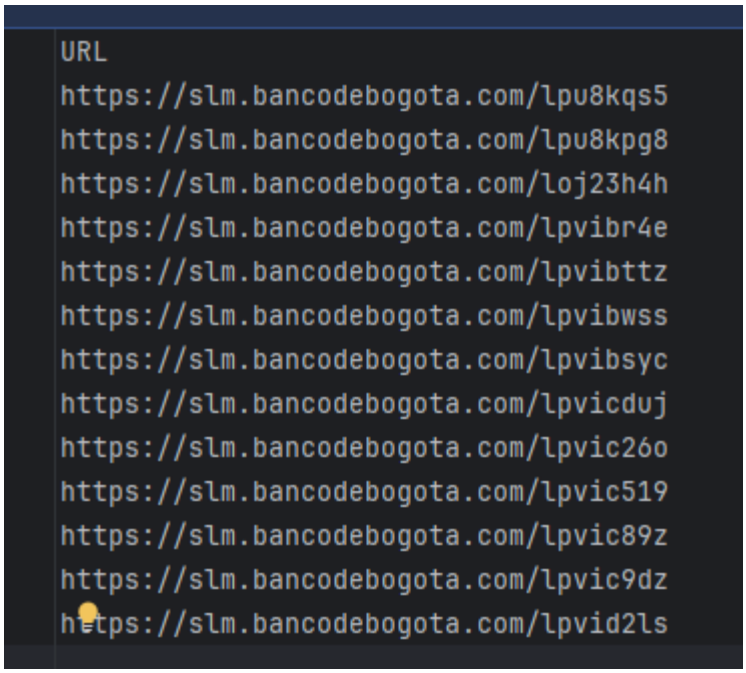


Figura 23. Links exitosos CC

Nota. Links exitosos de validación. (s.f.). Obtenido de Autor

Figura 24. Links Erróneos TC

Nota. Links fallidos de validación. (s.f.). Obtenido de Autor

Con esto se verifica el correcto funcionamiento y disponibilidad a la entidad para su uso de manera privada.

El flujo final tanto para CC (Crediconvenio) quedaría finalmente como en la Fig 25 y para TC (Tarjeta de crédito) como en la Fig 26.

Figura 25. Flujo del proceso de CC

Nota. Esquema de flujo de CC. (s.f.). Obtenido de Autor

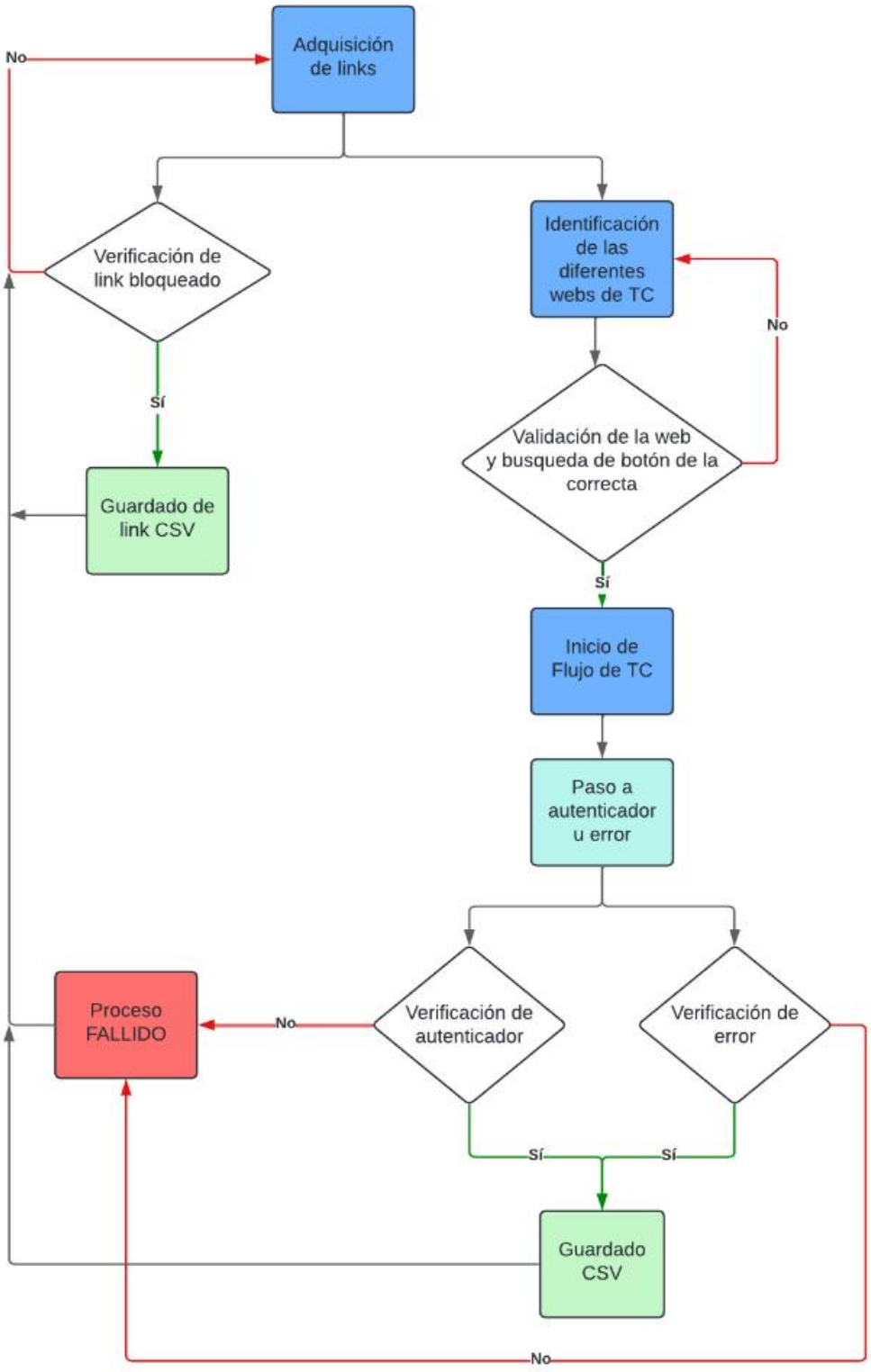


Figura 26. Flujo del proceso de TC

Nota. Esquema de flujo TC. (s.f.). Obtenido de Autor

11 Otros aportes realizados

Durante el desarrollo tanto del proyecto como de la pasantía se realizaron otros aportes como lo son la verificación de diferentes implementaciones dentro del Banco donde se tuvo en cuenta la correcta funcionalidad de cada uno de estos dentro de los ambientes de pruebas de la compañía los cuales son:

Lista de shortlinks - automatización validación QR B2B-218

Quiero guardar la lista de shortlinks para revisión de los recursos generados.

Por medio de curls se verifico la obtención de los recursos donde al momento de realizar la petición se esperaba un CSV con todos los links correspondientes a cada uno.

```
curl --location 'https://api-ald-admin.labdigbdbstgae.com/ald-adn
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--data '[
   {
      "Code": "1888",
      "Name": "Tests",
      "ShortName": "Test",
      "SalePoints": [
         {
            "Code": "0001",
            "Name": "Test",
            "IdCoordinator": "10300303"
         },
         {
            "Code": "0001",
            "Name": "Test",
            "IdCoordinator": "10300303"
         }
      ]
   }
]'
```

Figura 27. Curl Creación de recursos

Nota. Curl de creación de recursos. (s.f.). Obtenido de Autor

Figura 28. Archivo CSV

Nota. Bucket para obtención de archivo CSV. (s.f.). Obtenido de Autor

Automatización flujo CC Desembolso Mixto B2B-173

Como banco quiero automatizar el flujo de creación de recursos para crediconvenio desembolso mixto consultando las políticas existentes.

Por medio de un archivo CSV se generan los diferentes recursos necesarios para los aliados de crediconvenio de la entidad, se realizó la verificación de los logs donde se observaría la creación de los recursos con diferentes aliados y generando los posibles errores al momento de llenar los datos en el CSV.

Figura 29. Recursos Crediconvenio

Nota. CloudWatch de recursos. (s.f.). Obtenido de Autor

Flujo de creación automática de aliados se va por un flujo incorrecto B2B-

Al hacer el cargue de la plantilla de creación de aliados, esta se va por un flujo incorrecto de creación de recursos QR.

Al realizar el cargue de la creación de un nuevo aliado por medio de una plantilla este tomaba un flujo incorrecto y esta creación del aliado no se estaba realizando, por ende, los desarrolladores realizaron la corrección de este y se comprobó el correcto funcionamiento de la plantilla donde se identifican los logs correspondientes a la landa a la cual apuntaban Fig 29, validando la plantilla con diferentes inserciones y posibles errores para verificar esta.

Figura 30. Logs cargue de plantilla

Nota. CloudWatch de cargue de plantilla. (s.f.). Obtenido de Autor

BD unificado: Desplegar componente allied manager

Como Banco quiero desplegar el componente allied manager, el cual es la entrada al API de TVS para pasar el apuntamiento al componente SS (pasar el código que está ejecutándose de

la lambda a fargate). Esto se realiza con el objetivo de eliminar la dependencia que se tiene actualmente con TVS cuando se requiere tocar en producción la información de BD de los aliados para soporte, eliminar esta dependencia nos va a permitir ser más rápidos en la respuesta de los casos y operación de los aliados con los recursos/producto que tengan del Banco

Para esta aplicación se migro toda una base de datos de TVS el cual es la información principal que se usa para diversos equipos de desarrollo, y por medio de una nueva landa se apunta a este, donde se realizó la prueba de los diferentes curls los cuales apuntaban a la principal y se verifico el resultado de estos Fig 30.

```
{
    "name": "TESTS_10",
    "identificationNumber": "90136299",
    "identificationType": "NIT",
    "email": "testingpl@test.com",
    "phone": "********4484",
    "shortName": "test_01",
    "allyCode": "1998",
    "licenceExpiration": "2024/12/26 23:59:59",
    "category": {
        "code": "2550",
        "description": "Prueba Test"
    },
    "salePoints": {
        "0": {
            "name": "PV Bogota",
            "code": "0001",
            "type": "Physical",
            "email": "testingpl@test.com",
            "phone": "********4484",
            "address": "**********",
            "account": {
                "type": "Current",
                "number": "********"
            }
        },
        "1": {
            "name": "PV Medellin",
            "code": "0002",
            "type": "Physical",
            "email": "testingpl@test.com",
            "phone": "********4484",
            "address": "**********",
            "account": {
                "type": "Current",
                "number": "********"
            }
        },
        "2": {
            "name": "PV Cali",
            "code": "0003",
            "type": "Physical",
            "email": "testingpl@test.com",
            "phone": "********4484",
```


Figura 31. Logs de creación de aliados

Nota. CloudWatch de creación de recursos. (s.f.). Obtenido de Autor

Donde estas son unas de las diferentes labores realizadas en la pasantía, además de la modificación, actualización de datos, eliminación de datos e inserción de datos en bases de datos MySQL y NoSQL, donde esta labor era atención a solución en diferentes aliados del Banco de Bogotá.

12 Conclusiones

La librería Selenium de Python es útil para automatizar procesos manuales en páginas web, evitando tareas repetitivas y que consumen mucho tiempo. Además, mediante el uso de bibliotecas adicionales como Pytest, es posible documentar y detectar posibles fallos al implementar nuevas modificaciones en la página web.

Al realizar la automatización con Selenium, se deben considerar varios factores debido a su amplia funcionalidad y diversas capacidades. Es fundamental tener en cuenta los XPATH, ya que no siempre se identifican fácilmente o su ubicación puede no ser visible. Además, es crucial identificar correctamente el tipo de interacción necesaria, ya sea con botones, selectores u otros elementos. Si no se identifican adecuadamente, es probable que la automatización falle.

Los bots son de gran ayuda para realizar tareas manuales tediosas. En el caso de la comprobación masiva de enlaces, ejecutar estos flujos manualmente puede ser extremadamente tedioso y consumir mucho tiempo. Implementar bots para estas tareas no solo libera al equipo de una labor monótona, sino que también permite que el tiempo ahorrado se utilice en actividades de mayor impacto, mejorando así la eficiencia y los resultados del equipo.

Durante una pasantía, es esencial ser proactivo, autodidacta y responsable como profesional. A menudo, es necesario fortalecer los conocimientos en diversas áreas, ya que al realizar un desarrollo pueden surgir diferentes variables que uno podría considerar. La falta de conocimiento puede afectar al equipo de trabajo, por lo que es crucial poseer estas cualidades al integrarse a un equipo.

13 Bibliografía

[1] *¿Qué contiene XPath?* (s.f.). Obtenido de https://www.guru99.com/es/using-contains-sbiling-ancestor-to-find-element-in-selenium.html#:~:text=.quit()%3B%20%7D%20%7D-,XPath%20padre%20en%20Selenium,el%20elemento%20principal%20usando%20XPath.

[2] *¿Qué es el software de automatización?* . (s.f.). Obtenido de https://www.automationanywhere.com/la/rpa/automation-software

[3] *¿Qué es hardware y software?* (s.f.). Obtenido de https://edu.gcfglobal.org/es/informatica-basica/que-es-hardware-y-software/1/

[4] *¿Qué es la automatización?* (s.f.). Obtenido de https://www.redhat.com/es/topics/automation/whats-it-automation

[5] *¿Qué es Python?* . (s.f.). Obtenido de https://aws.amazon.com/es/what-is/python/

[6] *¿Qué es un archivo CSV?* . (s.f.). Obtenido de https://www.geeknetic.es/Archivo-CSV/que-es-y-para-que-sirve

[7] *¿Qué es un bot?* . (s.f.). Obtenido de https://aws.amazon.com/es/what-is/bot/#:~:text=Un%20bot%20es%20una%20aplicación,forma%20independiente%20sin%20intervención%20humana.

[8] *¿Qué es un bot? Tipos de bots y sus principales funciones*. (s.f.). Obtenido de https://www.godaddy.com/resources/es/crearweb/que-es-un-bot

[9] *¿Qué son las librerías de Python?* (s.f.). Obtenido de https://iddigitalschool.com/bootcamps/que-son-las-librerias-de-python/

[10] *¿Qué son los bots de software?* . (s.f.). Obtenido de https://www.automationanywhere.com/la/rpa/software-bots

[11] *Banco de Bogota landing CC*. (s.f.). Obtenido de https://slm.bancodebogota.com/lpu8kpg8

[12] *Banco de Bogota landing de error CC*. (s.f.). Obtenido de https://crediconveniodigital.bancodebogota.com.co/error

[13] *Banco de Bogota landing link no disponible*. (s.f.). Obtenido de https://slm.bancodebogota.com/ldewyftn/

[14] *Banco de Bogota landing TC*. (s.f.). Obtenido de https://digital.bancodebogota.com/tarjeta-credito/aliados/index.html?utm_source=claro_bajo_costo&utm_medium=referral&utm_campaign=claro_cla_asr57&lowcost=true

[15] *Banco de Bogota landing TC error*. (s.f.). Obtenido de https://tcdigital.bancodebogota.com/error/reject

[16] *Banco de Bogota landing TC Latam*. (s.f.). Obtenido de https://digital.bancodebogota.co/tarjeta-credito/landing-latampass/index.html

[17] *Banco de Bogota landing TC movistar*. (s.f.). Obtenido de https://digital.bancodebogota.com/tarjeta-credito/movistar/index.html

[18] *Bots, las herramientas omnipresentes en la era digital actual*. (s.f.). Obtenido de https://www.itmastersmag.com/noticias-analisis/bots-las-herramientas-omnipresentes-en-la-era-digital-actual/

[19] *Cómo codificar un scraping Bot con Selenium y Python*. (s.f.). Obtenido de https://www.freecodecamp.org/espanol/news/como-codificar-un-scraping-bot-con-selenium-y-python/

[20] *Cómo comprobar si «URL existe»*. (s.f.). Obtenido de https://uproc.io/blog/es/how-to-check-if-url-exists

[21] *Comprobar la existencia de enlaces rotos*. (s.f.). Obtenido de https://www.site24x7.com/es/tools/comprobar-de-enlaces.html

[22] *El impacto de las nuevas tecnologías en el empleo en México*. (s.f.). Obtenido de https://www.redalyc.org/journal/118/11858906002/html/#B32

[23] *Haz crecer tu negocio con datos optimizados*. (s.f.). Obtenido de https://uproc.io/#/?como-tratar-tus-datos-ii-validacion

[24] *Lenguajes de programación: qué son y cómo funcionan*. (s.f.). Obtenido de https://www.pixartprinting.es/blog/lenguajes-de-programacion/

[25] *Los bots toman cada día un lugar de mayor importancia en las empresas*. (s.f.). Obtenido de https://www.americaeconomia.com/los-bots-toman-cada-dia-un-lugar-de-mayor-importancia-en-las-empresas

[26] ofrece?, A. ¿. (s.f.). Obtenido de https://www.redhat.com/es/topics/automation

[27] *Programación orientada a objetos*. (s.f.). Obtenido de https://www.ibm.com/docs/es/spss-modeler/saas?topic=language-object-oriented-programming

[28] *Pytest*. (s.f.). Obtenido de https://docs.pytest.org/en/8.2.x/

[29] *Python*. (s.f.). Obtenido de https://www.python.org/about/

[30] *Qué es un lenguaje de programación*. (s.f.). Obtenido de https://openwebinars.net/blog/que-es-un-lenguaje-de-programacion/

[31] *Selenium and WebDriver*. (s.f.). Obtenido de https://www.electronjs.org/es/docs/latest/tutorial/using-selenium-and-webdriver#:~:text=ChromeDriver%20es%20un%20servidor%20independiente,equipos%20de%20Chromium%20y%20WebDriver.

[32] *Selenium Python*. (s.f.). Obtenido de https://selenium-python.readthedocs.io/installation.html

[33] *Selenium y la automatización de las pruebas*. (s.f.). Obtenido de https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/381#:~:text=Descripción,forma%20automática%20e%20iterativa%20posteriormente