AUTOMATIZACIÓN DE GUIAS EN SYSLIMA

Overskull

16/02/2025

Versión 1.0

Queda prohibido cualquier tipo de explotación y, en particular, la reproducción, distribución, comunicación pública y/o transformación, total o parcial, por cualquier medio, de este documento sin el previo consentimiento expreso y por escrito a overskull.

ÍNDICE

1. INTRODUCCIÓN Y OBJETIVOS

1.1 Contexto

La eficiencia en el sector logístico depende directamente de la velocidad con la que se procesa la información. Este proyecto se desarrolla para optimizar el registro en el portal SYSLIMA, una plataforma esencial que, debido a sus múltiples filtros y validaciones, suele generar demoras en el despacho manual de encomiendas durante las horas punta.

1.2 Propósito

El objetivo es transformar el llenado de datos, que actualmente es una tarea mecánica y lenta, en un proceso automático y fluido. Con esta automatización se busca:

- Liberar al personal de la carga operativa para que se enfoque en la calidad de atención al cliente.

- Garantizar la precisión de la información, eliminando errores de digitación en datos críticos.

- Acelerar el despacho, permitiendo que el sistema procese la información en una fracción del tiempo que toma hacerlo manualmente.

2. ALCANCE TÉCNICO

2.1 Automatización Operativa

La solución cubre el ciclo de vida integral del registro de la guía, eliminando la navegación manual redundante. Este proceso incluye:

1. **Gestión de Autenticación:** Protocolo de acceso seguro al portal mediante la inyección automatizada de credenciales.
2. **Navegación Estructurada:** Acceso directo a módulos profundos como "Generales" y "Orden de Servicio", superando la latencia de los menús desplegables.
3. **Sincronización Inteligente:** El bot utiliza algoritmos de espera (pausas técnicas) calculados para alinearse con los tiempos de respuesta del servidor de Shalom. Esto asegura que los datos del remitente y destinatario se carguen completamente desde la base de datos antes de proceder, evitando registros incompletos o errores de validación de DNI.

2.2 Gestión de Impresión

El bot está diseñado para operar en entornos multiventana, una de las fases más críticas y propensas a errores del proceso manual:

1. **Detección de Contexto:** Capacidad de identificar y transferir el control (switch) a ventanas emergentes de forma aislada, permitiendo que el script interactúe con el PDF sin interferir con la pestaña principal.
2. **Protocolo de Seguridad:** Gestión automatizada de la clave maestra. El bot no solo ingresa los dígitos, sino que gestiona la validación y confirmación en dos pasos dentro del modal de seguridad, garantizando que la generación del código final sea exitosa y libre de bloqueos visuales o cierres inesperados de la sesión.

2.3 Puntos de Control

Se ha implementado una arquitectura de automatización asistida. Este modelo permite que el bot se detenga estratégicamente durante la selección del destino, otorgando al operador el control

total sobre la ruta logística. Esta pausa crítica garantiza que la velocidad del bot no comprometa la toma de decisiones humanas en variables geográficas o agencias específicas, logrando una sinergia perfecta entre la precisión robótica y la flexibilidad del transporte logístico.

3. COMPONENTES Y DATA DEL PROCESO

3.1 Entradas (Inputs)

La integridad de la automatización reside en tres fuentes de datos fundamentales:

1. **Credenciales de Acceso:** Gestión de perfiles institucionales para la autenticación en el portal.
2. **Identificadores de Identidad:** Procesamiento dinámico de registros DNI/RUC para la consulta de bases de datos.
3. **Clave Maestra de Seguridad:** Uso de una secuencia predefinida (9191) para la autenticación en el módulo de firma digital y emisión de documentos, estandarizando la seguridad en cada guía generada.

3.2 Reglas de Negocio

El bot incorpora un protocolo de sincronización asíncrona. Para mitigar errores de tipo "Nombre no encontrado" o "Datos no cargados", se han programado pausas técnicas de entre 6 y 7 segundos. Este tiempo es indispensable para permitir que los servidores de Shalom completen la búsqueda en el padrón electoral o SUNAT, asegurando que los nombres de remitente y destinatario se reflejen correctamente antes de proceder con el registro del teléfono.

3.3 Mapeo de Selectores

Se han auditado y extraído rutas de navegación precisas (XPaths) y selectores de identidad (IDs) para cada nodo del formulario. Al priorizar identificadores estables como swal-input y telefonoEntrega, el bot adquiere "visión técnica", permitiéndole ignorar ventanas emergentes publicitarias o banners distractores, garantizando que la inserción de datos se realice exclusivamente en los campos correctos.

4. FLUJO OPERATIVO PASO A PASO

4.1 Navegación Inicial

El ciclo inicia con el despliegue del motor de navegación y la autenticación cifrada. Tras superar el acceso, el bot ejecuta una secuencia de búsqueda en el árbol de menús: despliega el panel lateral, localiza el nodo de "Generales" y activa el formulario de "Orden de Servicio". Este flujo reduce el tiempo de búsqueda manual a menos de 2 segundos.

4.2 Llenado de Formulario

La carga de información se ejecuta de manera modular para evitar colisiones de datos:

1. **Bloque de Identidad:** Registro y validación secuencial del remitente y destinatario.
2. **Bloque de Configuración:** El bot utiliza comandos de JavaScript (JS Executor) para seleccionar instantáneamente el método de pago y el tipo de paquete (Documentos). El uso de JS permite interactuar con elementos que a veces quedan ocultos o bloqueados por otros componentes de la página.

4.3 Finalización

Al completar el llenado, se activa el disparador de impresión. El sistema realiza una transferencia de foco a la pestaña del PDF, donde el bot gestiona el Módulo de Seguridad. En esta fase final, ingresa la clave en los campos de entrada y los repite en los de verificación con precisión matemática, culminando el proceso al accionar el botón de generación de código, dejando la guía lista para su distribución.

5. EVIDENCIAS TÉCNICAS (CAPTURA DE CÓDIGO Y SISTEMA)

5.1 Inspección de elementos DOM

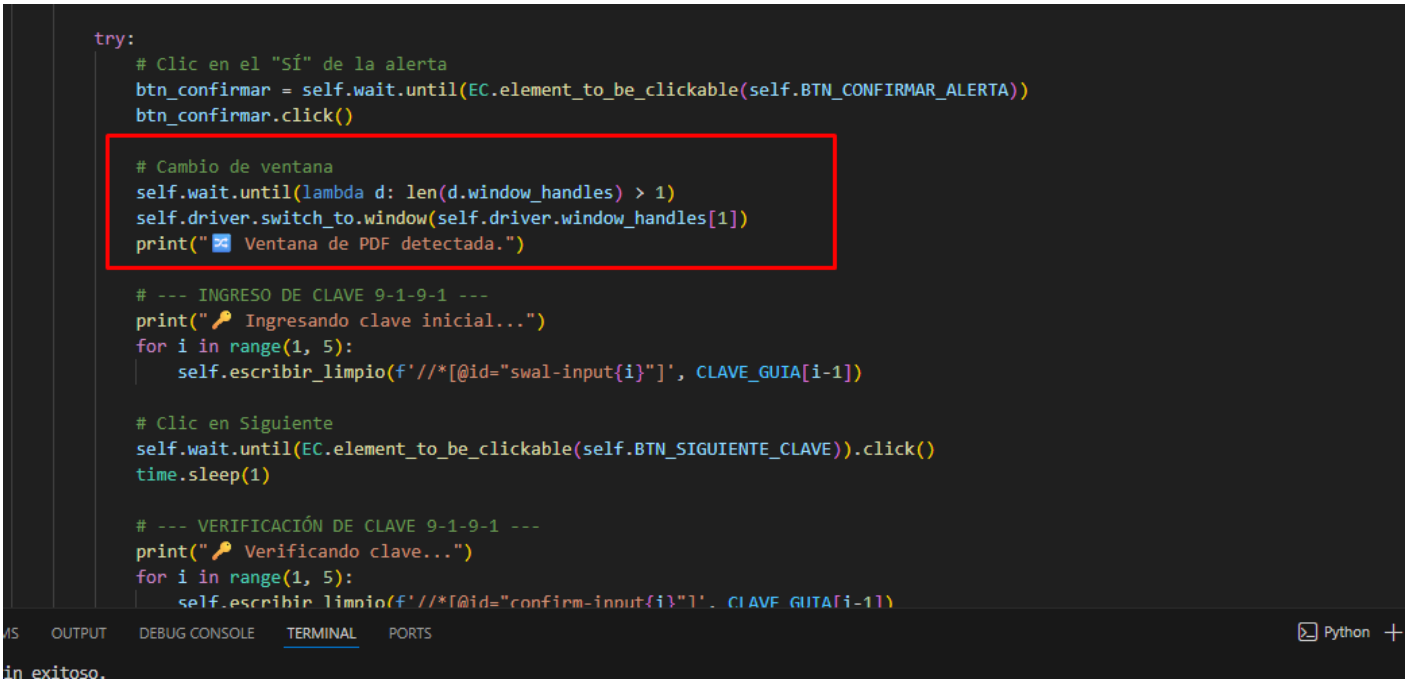
```
44
45      # SELECTORES
46      INPUT_USER = (By.XPATH, '//*[@id="loginaction"]/div/div[1]/input')
47      INPUT_PASS = (By.XPATH, '//*[@id="password-text"]')
48      BTN_LOGIN = (By.XPATH, '//*[@id="boton-submit"]')
49      BTN_HAMBURGUESA = (By.XPATH, '/html/body/div[3]/section/header/div[1]')
50      MENU_GENERALES = (By.XPATH, '/html/body/div[3]/section/div[2]/nav/ul/li[2]/a/span[2]')
51      MENU_ORDEN_SERVICIO = (By.XPATH, '/html/body/div[3]/section/div[2]/nav/ul/li[2]/ul/li[1]/a/span')
52
53      # CAMPOS FORMULARIO
54      XPATH_DNI_REM = '//*[@id="input_valida_campo2"]'
55      XPATH_TELF_REM = '//*[@id="input_telf_rem"]'
56      XPATH_DNI_DEST = '//*[@id="input_valida_campo3"]'
57      XPATH_TELF_DEST = '//*[@id="telefonoEntrega"]'
58
59      # IMPRESIÓN
60      BTN_IMPRIMIR_GUIA = (By.ID, "imprimePDF")
61      BTN_CONFIRMAR_ALERTA = (By.XPATH, '/html/body/div[5]/div/div[3]/button[1]')
62
63      # NUEVOS BOTONES MÓDULO PDF
64      BTN_SIGUIENTE_CLAVE = (By.XPATH, "//button[contains(text(), 'Siguiente')]")
65      BTN_CREAR_CODIGO = (By.XPATH, "//button[contains(text(), 'Crear codigo')]")
```

A través de herramientas de desarrollo, se capturó la jerarquía de los elementos swal-input. Estas

capturas demuestran que el bot interactúa con campos que tienen comportamientos automáticos (saltos de foco), los cuales fueron replicados en el código.

5.2 Lógica de Interacción con Ventanas

Uno de los mayores retos técnicos fue el manejo de múltiples pestañas. El siguiente fragmento de código muestra cómo el bot "salta" de la ventana principal a la de impresión:



5.3 Validación de Claves y Verificación

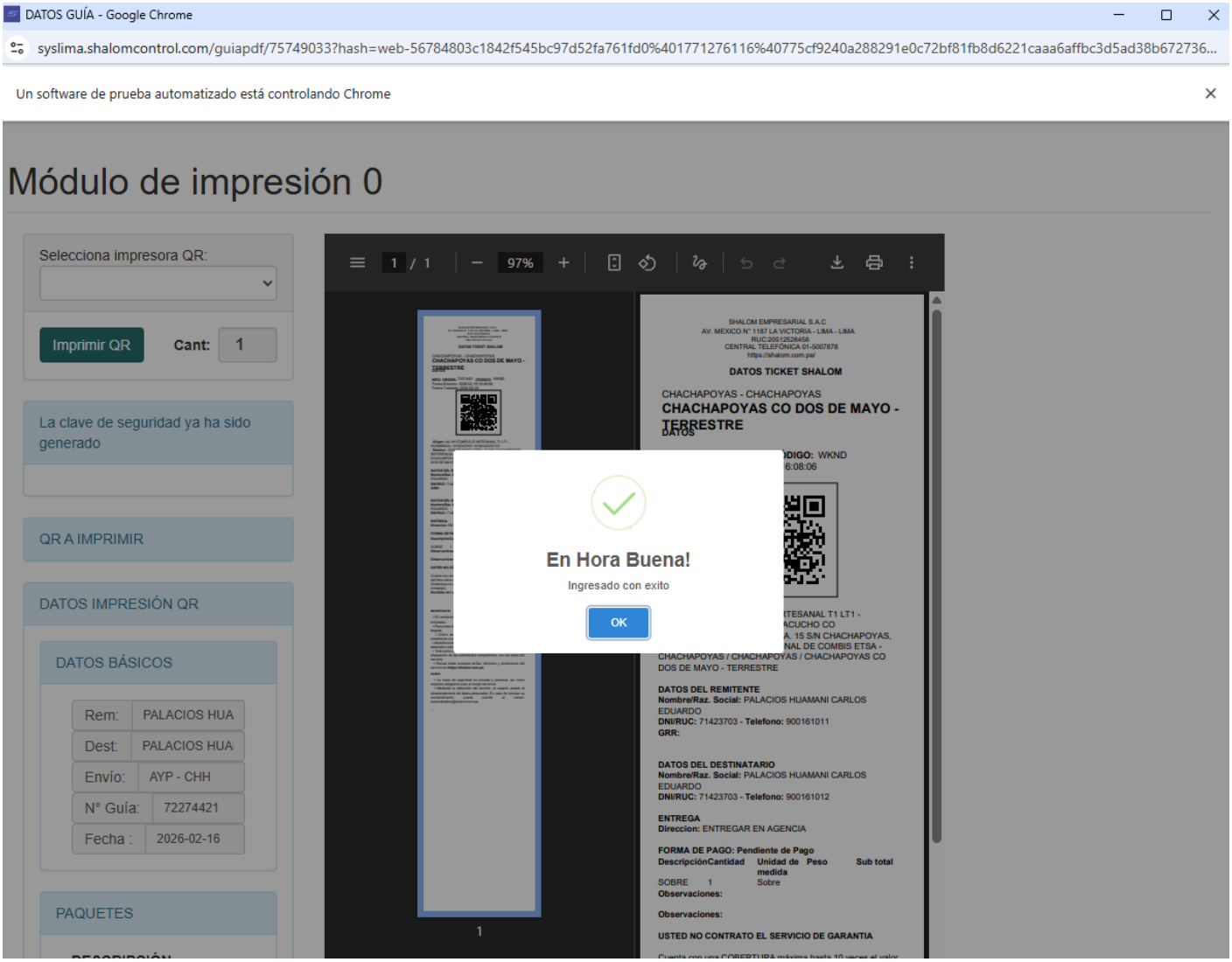
Se documenta el proceso de llenado doble (ingreso y confirmación). El bot replica el comportamiento humano ingresando los dígitos uno a uno para evitar bloqueos del sistema.

```
            self.wait.until(EC.element_to_be_clickable(self.BTN_SIGUIENTE_CLAVE)).click()
            time.sleep(1)

            # --- VERIFICACIÓN DE CLAVE 9-1-9-1 ---
            print("🔑 Verificando clave...")
            for i in range(1, 5):
                self.escribir_limpio(f'//*[@id="confirm-input{i}"]', CLAVE_GUIA[i-1])

            # --- BOTÓN FINAL ---
            print("🟢 Creando código final...")
            btn_crear = self.wait.until(EC.element_to_be_clickable(self.BTN_CREAR_CODIGO))
            btn_crear.click()
            print("✅ ¡GUÍA COMPLETADA!")

        except Exception as e:
            print(f"❌ Error en el módulo final: {e}")

        while True: time.sleep(10)

if __name__ == "__main__":
    bot = ShalomControlBot()
    try:
        bot.realizar_login()
        bot.navegar_a_orden()
        bot.completar_orden("71423703")
    except Exception as e:
        print(f"❌ Error: {e}")
```

Descripción: Interfaz final del módulo de generación de código antes de la ejecución del comando "Crear código".

5.4 Resultado de Ejecución Exitosa

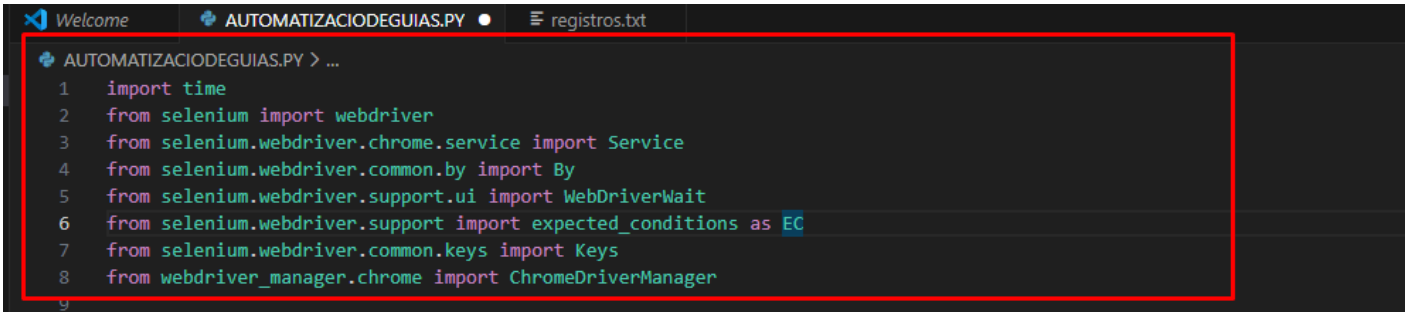
Se presenta la confirmación visual de que la guía ha sido procesada correctamente por el bot.



5.5 Proceso del codigo:

Paso 1: Inicialización y Configuración del Entorno

El script comienza configurando el navegador a través de `webdriver.ChromeOptions()`.



- **Lógica:** Se utiliza el comando `detach` para que la ventana permanezca abierta tras la ejecución. Se instancia el `WebDriverWait`, estableciendo un tiempo de espera preventivo de **25 segundos** para manejar la latencia del servidor.

Paso 2: Autenticación y Navegación Dinámica

El bot accede a la URL de login e inyecta las credenciales.

```
# ==========================================
# CONFIGURACIÓN
# ==========================================
URL = "https://syslima.shalomcontrol.com/login"
USUARIO = "70900128"
PASSWORD = "Prueba12."
```

- **Lógica:** Tras el ingreso, el código busca el "menú hamburguesa" y realiza una navegación por niveles: `Generales > Orden de Servicio`. Se utiliza `EC.element_to_be_clickable` para asegurar que el bot no intente hacer clic antes de que el menú se despliegue visualmente.

Paso 3: Validación de Identidad y Sincronización

Esta es la fase crítica donde el bot carga los datos del remitente y destinatario.

```
# CAMPOS FORMULARIO
XPATH_DNI_REM = '//*[@id="input_valida_campo2"]'
XPATH_TELF_REM = '//*[@id="input_telf_rem"]'
XPATH_DNI_DEST = '//*[@id="input_valida_campo3"]'
XPATH_TELF_DEST = '//*[@id="telefonoEntrega"]'

# IMPRESIÓN
```

- **Lógica:** Después de ingresar el DNI, el código ejecuta un `time.sleep(7)`. Este retraso no es aleatorio; es el tiempo necesario para que el **script interno de Shalom** realice la petición al padrón (RENIEC/SUNAT). Si el bot escribiera el teléfono inmediatamente, la web borraría el dato al refrescar el nombre.

Paso 4: Inyección por JavaScript (Método de Pago y Paquete)

Para evitar errores de "elemento no interactuable", el bot utiliza `execute_script`.

```
# PAGO Y PAQUETE
self.wait.until(EC.element_to_be_clickable((By.ID, "idosTipoPago"))).click()
self.driver.find_element(By.XPATH, '//*[@id="idosTipoPago"]/option[3]').click()
self.wait.until(EC.element_to_be_clickable((By.ID, "btn_name_paquete"))).click()
time.sleep(2)
self.wait.until(EC.element_to_be_clickable((By.XPATH, '//*[@id="formDeclaracionJurada"]/div/div[1]/div/div/div[2]'))).click()
self.wait.until(EC.element_to_be_clickable((By.ID, "declaracion_btn-agregar"))).click()
```

- **Lógica:** En lugar de un clic físico, el bot le ordena al navegador cambiar el valor del selector de pago directamente en el código de la página. Esto garantiza que la opción "Entrega" y el tipo "Documentos" se seleccionen incluso si hay banners sobrepuestos.

Paso 5: Gestión de la Ventana Emergente (Módulo de Impresión)

Cuando se pulsa "Imprimir", el sistema abre una nueva pestaña.

```python
# PROCESO DE IMPRESIÓN
print("🚀 Iniciando impresión...")
time.sleep(2)
btn_pdf = self.wait.until(EC.element_to_be_clickable(self.BTN_IMPRIMIR_GUIA))
self.driver.execute_script("arguments[0].click();", btn_pdf)

try:
    # Clic en el "SÍ" de la alerta
    btn_confirmar = self.wait.until(EC.element_to_be_clickable(self.BTN_CONFIRMAR_ALERTA))
    btn_confirmar.click()

    # Cambio de ventana
    self.wait.until(lambda d: len(d.window_handles) > 1)
    self.driver.switch_to.window(self.driver.window_handles[1])
    print("🔀 Ventana de PDF detectada.")
```

- **Lógica:** El bot utiliza `self.driver.window_handles[1]`. El índice `[0]` es la página principal y el `[1]` es el PDF. Sin esta línea, el bot seguiría intentando actuar sobre la página de inicio, resultando en un error de ejecución.

Paso 6: Aplicación de Clave y Cierre de Ciclo

El bot enfrenta dos formularios de clave (`swal-input` y `confirm-input`).

```python
        time.sleep(1)

        # --- VERIFICACIÓN DE CLAVE 9-1-9-1 ---
        print("🔑 Verificando clave...")
        for i in range(1, 5):
            self.escribir_limpio(f'//*[@id="confirm-input{i}"]', CLAVE_GUIA[i-1])

        # --- BOTÓN FINAL ---
        print("🟢 Creando código final...")
        btn_crear = self.wait.until(EC.element_to_be_clickable(self.BTN_CREAR_CODIGO))
        btn_crear.click()
        print("✅ ¡GUÍA COMPLETADA!")
```

- **Lógica:** Mediante un bucle `for`, el bot recorre los 4 campos de entrada de forma secuencial. Al finalizar la segunda validación con la clave `9191`, localiza el botón mediante un selector de texto exacto (`contains(text(), 'Crear codigo')`) para disparar la generación final del documento.

6. GESTIÓN DE RIESGOS Y BUENAS PRÁCTICAS

6.1 Riesgos Detectados

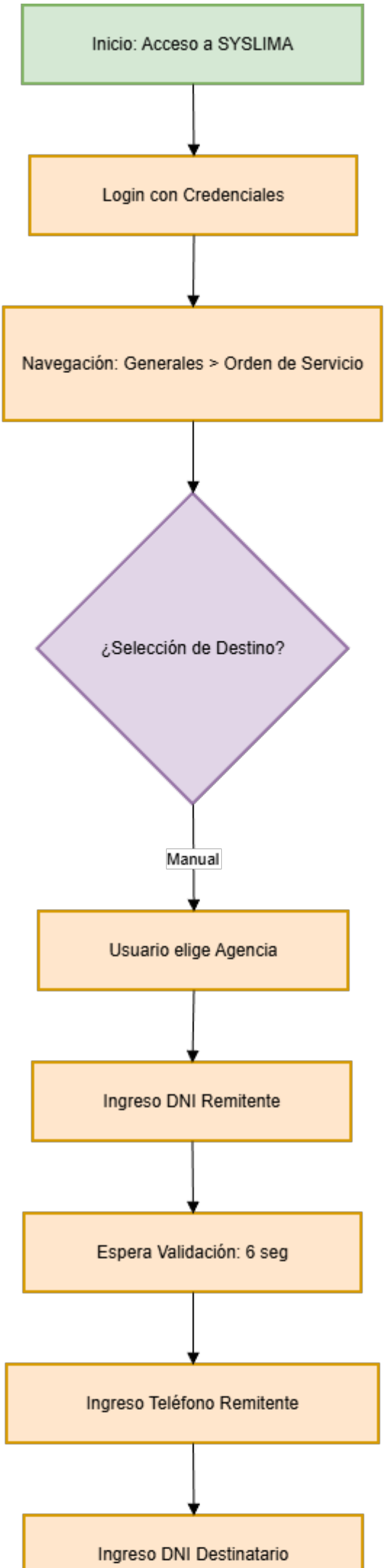
1. **Inestabilidad de Selectores:** El riesgo crítico es la actualización del portal SYSLIMA. Si el equipo técnico de Shalom modifica los IDs o XPaths de los campos, el bot perderá la

capacidad de localizar elementos, causando interrupciones en el flujo.
2. **Latencia de Red:** Conexiones inestables pueden retrasar la carga de elementos dinámicos. Esto obligaría a reajustar los tiempos de espera (Wait Times) para evitar que el bot intente interactuar con campos que aún no son visibles.
3. **Seguridad y Bloqueos:** El uso de automatización a velocidades no humanas puede ser detectado por sistemas anti-bot, resultando en bloqueos temporales de la IP o la cuenta si no se respetan las pausas programadas.

6.2 Recomendaciones

1. **Mantenimiento Preventivo:** Realizar una auditoría semanal de los XPaths principales para asegurar que el mapeo siga vigente tras cada actualización del sitio web.
2. **Actualización de Entorno:** Mantener el navegador Chrome y su respectivo WebDriver sincronizados en la misma versión para evitar fallos de compatibilidad en el arranque del script.
3. **Protocolo de Validación:** Es imperativo no reducir las pausas de 6 a 7 segundos. Estos tiempos actúan como un "seguro" que permite la sincronización con los servidores de identidad y simula un comportamiento de navegación orgánica para evitar sanciones del portal.
4. **Monitoreo de Logs:** Registrar las fallas frecuentes para identificar si los errores son causados por la web de Shalom o por caídas locales de internet.

7.Diagrama de Flujo


Inicio: Acceso a SYSLIMA
Login con Credenciales
Navegación: Generales > Orden de Servicio
¿Selección de Destino?
Manual
Usuario elige Agencia
Ingreso DNI Remitente
Espera Validación: 6 seg
Ingreso Teléfono Remitente
Ingreso DNI Destinatario

8.CODIGO COMPLETO DOCUMENTADO

```
import time
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from webdriver_manager.chrome import ChromeDriverManager

#
============================================================
====================
# CONFIGURACIÓN Y PARÁMETROS GLOBALES
#
============================================================
====================
URL_LOGIN = "https://syslima.shalomcontrol.com/login"
USER_DATA = {"user": "70900128", "pass": "Prueba12."}
CONTACTOS = {"remitente": "900161011", "destinatario": "900161012"}
CLAVE_MAESTRA = "9191"

class ShalomBot:
    def __init__(self):
        """Inicializa el navegador con opciones para mantener la sesión abierta."""
        print("□□ Iniciando motor de automatización...")
        options = webdriver.ChromeOptions()
        options.add_argument("--disable-notifications")
        options.add_experimental_option("detach", True) # Evita que Chrome se cierre al terminar

        self.driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()),
options=options)
        self.driver.maximize_window()
        self.wait = WebDriverWait(self.driver, 25) # Tiempo de espera para carga de elementos

    def escribir_campo(self, xpath, texto, limpiar=True):
        """Limpia un campo de texto e ingresa la información de forma segura."""
        try:
            elemento = self.wait.until(EC.element_to_be_clickable((By.XPATH, xpath)))
            if limpiar:
                elemento.click()
                elemento.send_keys(Keys.CONTROL + "a")
                elemento.send_keys(Keys.BACKSPACE)
```

```
            elemento.send_keys(texto)
            return True
        except Exception as e:
            print(f"⚠ Error al escribir en {xpath}: {e}")
            return False

    def login(self):
        """Realiza la autenticación en el portal SYSLIMA."""
        print("□□ Accediendo al sistema...")
        self.driver.get(URL_LOGIN)
        self.escribir_campo('//*[@id="loginaction"]/div/div[1]/input', USER_DATA["user"])
        self.escribir_campo('//*[@id="password-text"]', USER_DATA["pass"])
        self.driver.find_element(By.ID, "boton-submit").click()
        print("□  Autenticación exitosa.")

    def navegar_a_formulario(self):
        """Navega a través del menú lateral hasta 'Orden de Servicio'."""
        time.sleep(2)
        # Clic en menú hamburguesa
        self.wait.until(EC.element_to_be_clickable((By.XPATH,
'/html/body/div[3]/section/header/div[1]'))).click()
        # Clic en sección Generales
        self.wait.until(EC.element_to_be_clickable((By.XPATH,
"//span[contains(text(),'Generales')]"))).click()
        # Clic en Orden de Servicio
        self.wait.until(EC.element_to_be_clickable((By.XPATH, "//span[contains(text(),'Orden de
servicio')]"))).click()
        print("□□ Formulario de orden cargado.")

    def procesar_guia(self, dni_cliente):
        """Ejecuta el flujo completo de llenado de la guía."""

        # --- PASO 1: CONTROL HÍBRIDO ---
        print("\n□□ ESPERANDO SELECCIÓN DE DESTINO (Hazlo manualmente en el navegador)...")
        input("□□ Una vez seleccionado el destino, presiona ENTER aquí para continuar...")

        # --- PASO 2: DATOS DEL REMITENTE ---
        print(f"□□ Remitente: Ingresando DNI {dni_cliente}")
        self.escribir_campo('//*[@id="input_valida_campo2"]', dni_cliente)
        time.sleep(6) # Espera técnica para validación de base de datos
        self.escribir_campo('//*[@id="input_telf_rem"]', CONTACTOS["remitente"])

        # --- PASO 3: DATOS DEL DESTINATARIO ---
        print(f"□□ Destinatario: Ingresando DNI {dni_cliente}")
        self.escribir_campo('//*[@id="input_valida_campo3"]', dni_cliente)
        time.sleep(7) # Espera técnica para carga de identidad
```

```
        self.escribir_campo('//*[@id="telefonoEntrega"]', CONTACTOS["destinatario"])

        # --- PASO 4: CONFIGURACIÓN DE PAGO Y PAQUETE ---
        print("□□ Configurando pago (Entrega) y tipo de producto...")
        # Selección de Pago
        self.wait.until(EC.element_to_be_clickable((By.ID, "idosTipoPago"))).click()
        self.driver.find_element(By.XPATH, '//*[@id="idosTipoPago"]/option[3]').click() # Opción
'Entrega'
        # Agregar Producto
        self.driver.find_element(By.ID, "btn_name_paquete").click()
        time.sleep(1.5)
        self.wait.until(EC.element_to_be_clickable((By.XPATH,
'//*[@id="formDeclaracionJurada"]/div/div[1]/div/div/div[2]'))).click() # Documentos
        self.driver.find_element(By.ID, "declaracion_btn-agregar").click()

        # --- PASO 5: IMPRESIÓN Y SEGURIDAD ---
        print("□□ Generando impresión...")
        time.sleep(2)
        btn_imprimir = self.driver.find_element(By.ID, "imprimePDF")
        self.driver.execute_script("arguments[0].click();", btn_imprimir)

        # Confirmar cuadro de diálogo "SÍ"
        self.wait.until(EC.element_to_be_clickable((By.XPATH,
'/html/body/div[5]/div/div[3]/button[1]'))).click()

        # Cambio a la ventana emergente del PDF
        self.wait.until(lambda d: len(d.window_handles) > 1)
        self.driver.switch_to.window(self.driver.window_handles[1])
        print("□□ Foco transferido al Módulo de Impresión.")

        # --- PASO 6: CREACIÓN DE CÓDIGO DE SEGURIDAD ---
        print(f"□□ Aplicando clave maestra: {CLAVE_MAESTRA}")
        # Bloque de ingreso inicial (swal-input)
        for i, num in enumerate(CLAVE_MAESTRA, 1):
            self.escribir_campo(f'//*[@id="swal-input{i}"]', num, limpiar=False)

        self.driver.find_element(By.XPATH, "//button[contains(text(), 'Siguiente')]").click()
        time.sleep(1)

        # Bloque de verificación (confirm-input)
        for i, num in enumerate(CLAVE_MAESTRA, 1):
            self.escribir_campo(f'//*[@id="confirm-input{i}"]', num, limpiar=False)

        # Clic final
        self.driver.find_element(By.XPATH, "//button[contains(text(), 'Crear codigo')]").click()
        print("□  PROCESO FINALIZADO CON ÉXITO.")
```

```
#
================================================================
====================
# EJECUCIÓN PRINCIPAL
#
================================================================
====================
if __name__ == "__main__":
    bot = ShalomBot()
    try:
        bot.login()
        bot.navegar_a_orden()
        bot.procesar_guia("71423703")
    except Exception as e:
        print(f"  Error crítico en la ejecución: {e}")
```

Revisión #4
Creado 2026-02-16 10:27:10 -05 por MARCO SISTEMAS
Actualizado 2026-02-16 16:50:56 -05 por MARCO SISTEMAS