Fundación Universitaria del Área Andina
Facultad de Ingenierías
Ingeniería de Sistemas - Modalidad Virtual
Pasantías Empresariales

AUTOMATIZACIÓN DE CASOS DE PRUEBA, PARA ASEGURAMIENTO DE CALIDAD EN EL DESARROLLO DE SOFTWARE, EN APLICACIÓN WORKMAGAER ED CLOUD®

Presentado Por:

Pablo Camilo Vásquez Segura

Colombia, Bogotá D.C., 2021

AREANDINA
Fundación Universitaria del Área Andina

Contenido

PROPUESTA DE AUTOMATIZACIÓN DE CASOS DE PRUEBA PARA ASEGURAMIENTO DE CALIDAD EN EL DESARROLLO DE SOFTWARE

1. Empresa Estrategias Documentales SAS

1.1 Software WM ED CLOUD®

es un sistema de gestión de documentos electrónicos de archivo (SGDEA) desplegado en la nube, cuyo propósito es capturar, almacenar, acceder, gestionar y organizar la información física y electrónica de una organización, cumpliendo con la normativa archivística vigente y los requisitos técnicos y funcionales de la MOREQ. Cuenta con una interfaz de usuario moderna, intuitiva y de fácil manejo que se adapta fácilmente a cualquier dispositivo (computador de escritorio, computador portátil, tableta o celular inteligente) facilitando el ingreso desde diferentes lugares.
WM ED Cloud, se encuentra desarrollado bajo la modalidad de servicio SaaS y cuenta con el acceso mediante suscripción mensual o anual.

Los términos y/o conceptos acá presentados, están tanto en español como en inglés, dado que la corriente anglosajona es la de mayor uso en esta disciplina y para facilitar su consulta y entendimiento, los mismos no son los únicos en el tema, solo fueron elegidos porque son los que más escuchamos, pero existen muchos otros que no quedaron incluidos en este glosario y lo ideal es que este documento continúe siendo alimentado con aquellos términos que consideremos se deben incluir para facilitar nuestra labor diaria dentro del software .

Esperamos que este glosario pueda aportar y aclarar algunos de los conceptos que escuchamos a diario en nuestra labor.

1.2 Misión

Es una organización dedicada a la integración y optimización de los procesos de gestión documental tales como la recepción, producción, distribución, trámite, organización, conservación, disposición final y consulta de información bajo esquemas de eficacia y eficiencia, conformada por un equipo humano competente.

1.3 Visión

Consolidarnos para el 2025 como una empresa innovadora y sostenible, garantizando la continuidad del negocio y su crecimiento en el mercado nacional e internacional, reconocida por nuestros colaboradores, aliados, clientes y proveedores en prestación de servicios y desarrollo de software de administración de documentos, información y flujos de trabajo electrónicos con altos estándares de calidad.

1.4 Ubicación

Estrategias documentales, cuenta con una sede principal ubicada en Sabaneta, Antioquia, en la dirección 47d-185 a, Cl. 80 Sur

1.5 Resumen empresarial

Estrategias documentales cuenta actualmente dentro de su portafolio, con un Software de Gestión Documental, Gestión de Información y Flujos de Trabajo. Es una plataforma integral, colaborativa y participativa que tiene como propósito distribuir y gestionar el conocimiento y la información de la empresa en formato electrónico. Con acceso remoto a través de internet, puede también desplegarse en cualquier dispositivo móvil y en todos los navegadores disponibles, actualmente se encuentra en proceso de desarrollo el software WorkManager ED Cloud® el cual ara parte de las plataformas en la nube, dejando a un lado las instalaciones de escritorio.

2. Introducción

La automatización de pruebas de software es una de las opciones más fascinantes para enfrentar un equilibrio habitual de entregas al cliente sin comprometer la calidad del producto software; esto se hace más necesario cuando ya se tiene un producto en un ambiente de producción y con una gran cantidad de usuarios que pueden verse afectados por un fallo. Existen diversos tipos y niveles de pruebas, entre ellos las pruebas de aceptación las cuales tienen especial importancia de cara a la conformidad del comportamiento externo del producto según las expectativas del cliente.

El análisis se realizó para la aplicación WorkManager ED Cloud®, la cual, dentro de su proyecto de desarrollo, cuenta con procesos definidos y documentados para el desarrollo de funcionalidades y la ejecución de pruebas de calidad sobre dicho software. Esa ejecución de pruebas de calidad es un proceso que se ejecuta manualmente haciendo que se tome mayor tiempo y que exista una mayor posibilidad de que se cometan errores durante su ejecución. Además del análisis realizado, para el desarrollo de la propuesta de ejecución de casos de pruebas automáticas, se pretende realizar simulación de ejecución automática de casos reales buscando involucrar las herramientas que fueron identificadas inicialmente.

La automatización puede prometer grandes beneficios siempre y cuando recordemos que “pruebas” no solo significa probar lo mismo una y otra vez hasta el agotamiento: esto conlleva a evaluar los resultados con respecto a las diferentes entradas para así ver que pruebas dar valor agregado y saber que herramienta a emplear ya que esta nunca hará el trabajo solas sin un buen entendimiento.

3. Objetivos

3.1 Objetivo General

Diseñar una propuesta para la automatización y ejecución de pruebas, sobre las funcionalidades desplegadas para la aplicación web WorkManager ED Cloud®, con el fin de reducir tiempo en su ejecución y logrando una mejor respuesta para la ejecución continua de pasos repetidos en las pruebas, despliegues de funcionalidades y mejoras.

3.2 Objetivo Especifico

Analizar el proceso de pruebas llevados hoy en día con relación al despliegue de funcionalidad que se aplican para el aplicativo WorkManager ED Cloud®, con el fin de identificar posibles mejoras mediante la automatización de casos de prueba.
Emplear herramientas de automatización como lo es Selenium, bajo el lenguaje de programación Python., posterior a ello plantear una propuesta de ejecución de casos de prueba automatizados en los desarrollos de WorkManager ED Cloud®, para que sea más eficiente en el proceso de aceptación y aprobación de los desarrollos realizados partiendo de Historias de Usuarios.

4. Proyecto Basado en la Automatización de Pruebas con Selenium y lenguaje de Programación Python

En este capítulo se presenta como funcionalidades a ser automatizadas de forma inicial, aquellos desarrollos a los cuales se les ha venido realizando un proceso de pruebas, integraciones con micro servicios desarrollados para la aplicación WorkManager ED Cloud.®.
Teniendo en cuenta lo descrito anteriormente, la automatización de los desarrollos estará basada en las funcionalidades actualmente desarrolladas:

- Inicio de Sesión en la aplicación WorkManager ED Cloud®
- Visualización de Entorno de trabajo de la aplicación WorkManager ED Cloud®
- Creación de Formularios en el aplicativo WorkManager ED Cloud®
- Creación de Campos en la Creación de Formularios en el aplicativo WorkManager ED Cloud®
- Radicación de Información en formularios creados
- Cargue de documentos con vinculación a la TRD.

5. Introducción a las pruebas del software

En este capítulo se describe el proceso de automatización de pruebas utilizando la propuesta de herramientas como Selenium y lenguaje de programación Python, el proceso comienza con definir que funcionalidades del proyecto WorkManager ED Cloud® serán las tenidas en cuenta para realizar el proceso de automatización, se debe tener en cuenta que las automatizaciones son basadas en pruebas funcionales, teniendo en cuenta los criterios de aceptación indicados en las historias de usuario desarrolladas por el PO del proyecto, estas historias de usuario han sido

aprobadas y validadas por el equipo scrum y sobre estas se realizaran las automatizaciones, las funcionalidades elegidas para el desarrollo de automatizaciones, corresponden a las funcionalidades de Login de usuario, contemplando sus diferentes validaciones a nivel de frontend , así como también posterior al Login el paso siguiente la visualización del dash board principal de la aplicación, seguido la funcionalidad generada para usuarios administradores correspondiente a creación de formularios, creación de campos en formularios creados y finalmente la radicación de estos formularios, para estas automatizaciones las funcionalidades nombradas anteriormente se validarían escenarios de pruebas sujetos a criterios de aceptación.

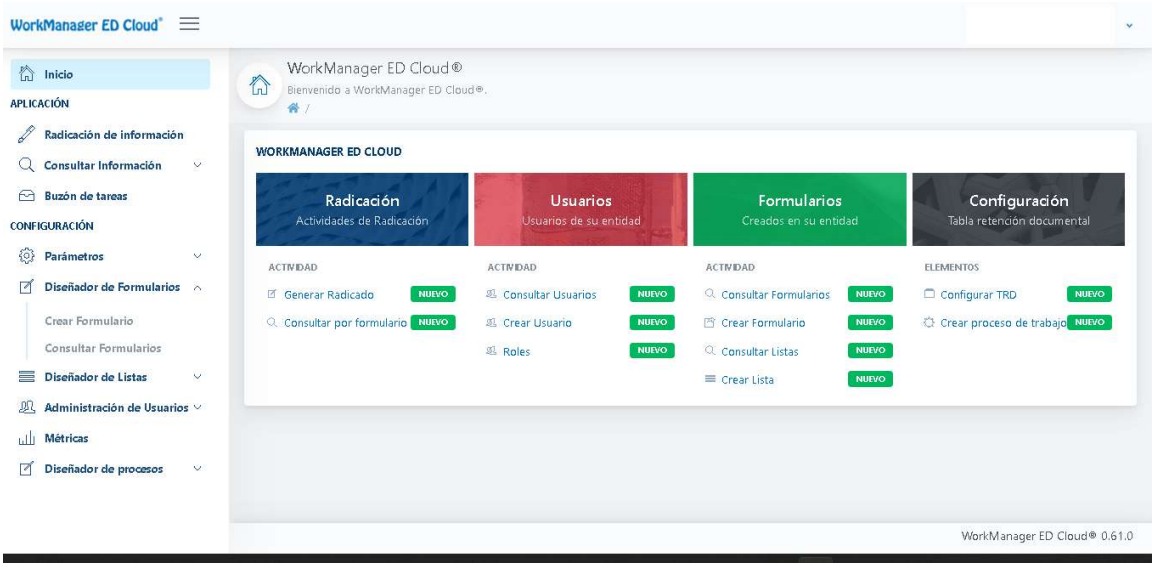


Ilustración 1 Dash Board Principal Aplicativo WorkManager ED Cloud®-Fuente Propia

6. Elección de los navegadores

Una de las principales preocupaciones para los desarrolladores WEB es el poder realizar desarrollos los cuales sean compatibles y realicen un correcto funcionamiento en el máximo número de navegadores web, de la misma manera la implementación de pruebas automatizadas busca la correcta ejecución de los script basándose en la implementación de extensiones que permiten a esta funcionalidad, poder ejecutarse de la mejor manera, por tal razón y teniendo encuesta lo anteriormente descrito se ha decidido y de acuerdo a los análisis de concurrencia en navegadores, implementar la automatización en el navegador Google Chrome. Nuestras pruebas de automatización correrán bajo el navegador Chrome contando con la extensión Selenium IDE y el driver ChromeDriver - WebDriver for Chrome.

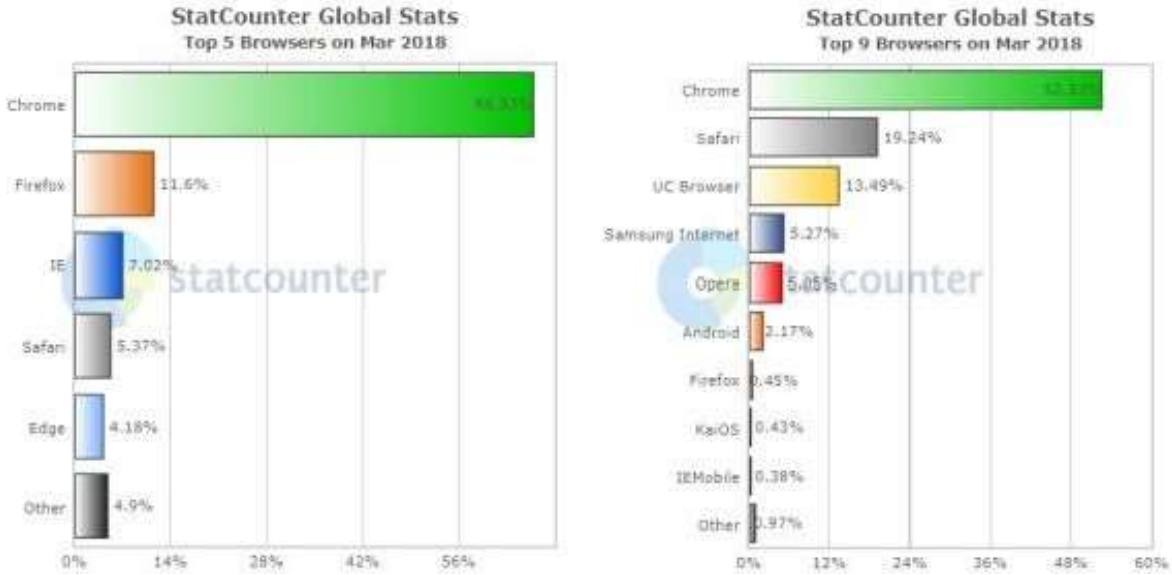


Ilustración 2: Estadísticas de uso de navegadores en el escritorio (izquierda) y móvil (derecha) a marzo de 2018, en porcentaje de usuarios. [13]

7. Herramientas para pruebas funcionales web

Como ya lo hemos mencionado, nuestro proyecto de automatización está basado en Selenium bajo el lenguaje de programación Python, para ello es necesario configurar nuestro entorno de trabajo, teniendo en cuenta esto se procederá a realizar las indicaciones correspondientes a la configuración pertinente.

7.1 Configuración de Entorno de Trabajo

Con el fin de realizar una instalación limpia de las herramientas requeridas para la elaboración y ejecución de script automatizados con Selenium y Python, es necesario instalar lo siguiente:

- Python en una versión superior a 3.6
- Selenium
- PyUnitReport (Librería la cual permite generar reportes en formato HTML.)
- Descarga de Driver de Selenium compatible con Google Chrome.
-

7.2 Instalación de Herramientas

1- Lo primero que se debe realizar es validar versión de Python que tenga instalada el equipo del usuario, para ello se debe ingresar a la consola de Windows y posteriormente escribir: Python –versión, de esta manera sabremos la versión que tenemos instalada, en caso de ser inferior a 3.6 o no contar con ella, se

puede realizar descarga e instalación de la misma, directamente de la página de Python : https://www.python.org/downloads/

Ilustración 3 Verificación Versión Python - Fuente Propia

2- Realizar instalación de Selenium en nuestro equipo, para ello se debe ingresar a la consola de comandos de nuestro navegador, posterior mente escribir lo siguiente: pip3 install Selenium

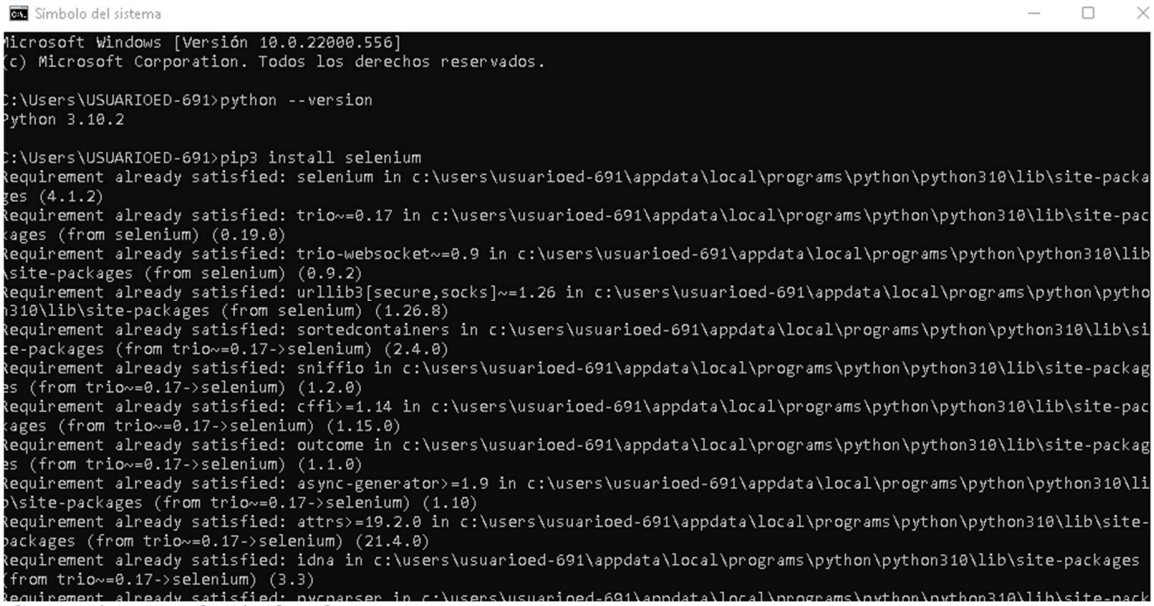


Ilustración 4 Instalación de Selenium- Fuente Propia

3- Por último, se debe realizar la instalación de la librería para generar reportes con Selenium, por tal razón y de la misma manera que la verificación de Python y la instalación de Selenium, ejecutamos nuestra consola de Windows y escribimos lo siguiente: pip3 install pyunitreport

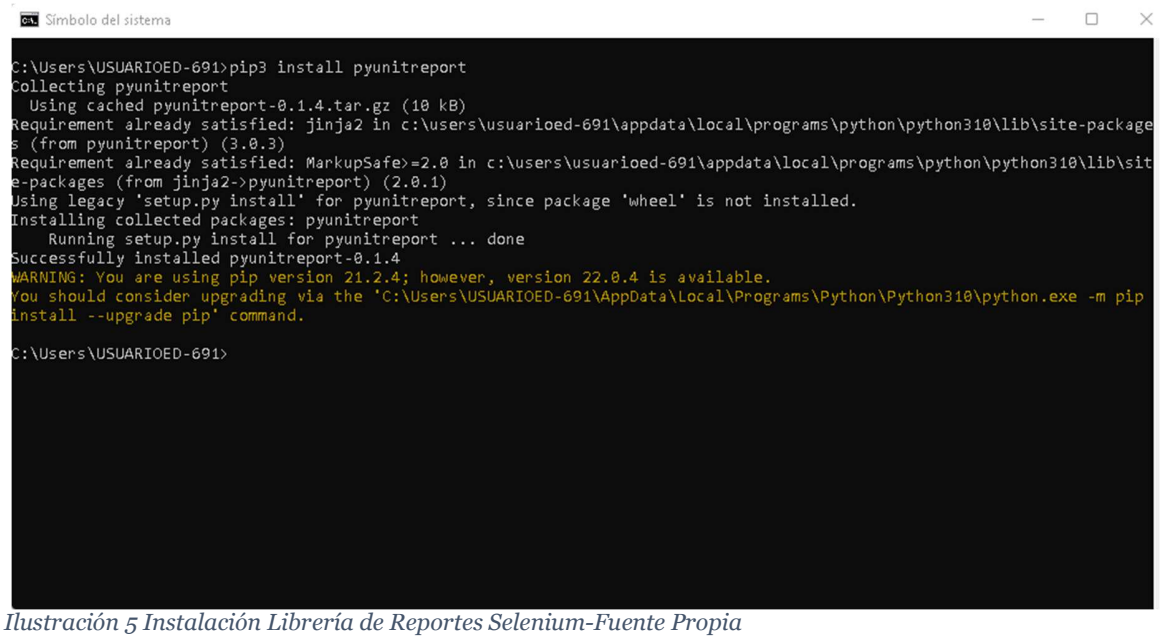


Ilustración 5 Instalación Librería de Reportes Selenium-Fuente Propia

4- Por último, paso, debemos realizar descarga del driver el cual permitirá la ejecución automática de nuestro script con el navegador, por tal razón se debe ingresar al sitio oficial de descarga: https://chromedriver.chromium.org/downloads posterior a ello, seleccionar la versión mas reciente y descargar el driver de acuerdo a nuestro sistema operativo.

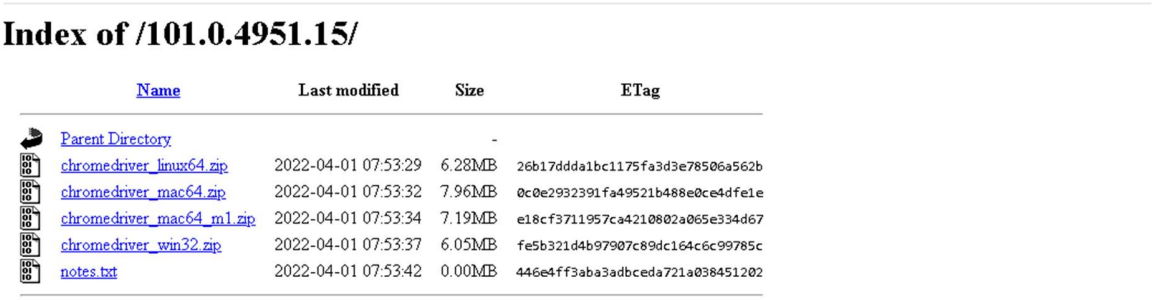


Ilustración 6 Versiones actualmente disponibles -Fuente Propia

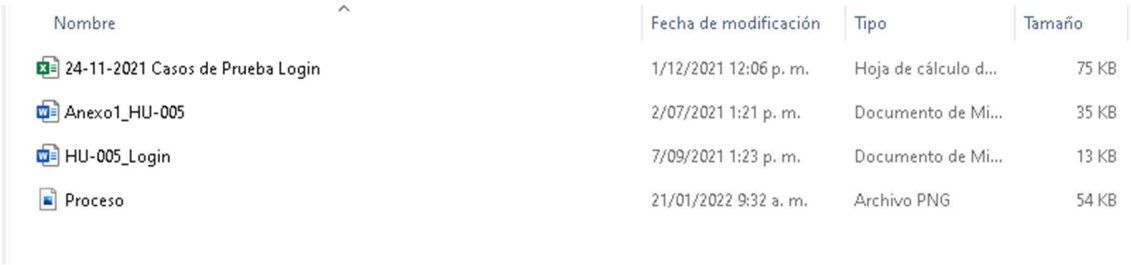
Nota: Posterior a la descarga del Driver es necesario alojarlo en una ruta la cual podamos recordar, esto dado a que, para las automatizaciones, esta ruta se llamara por medio de comentos para la ejecución del driver.

8. Actividad 1 Automatización de Funcionalidad Login

Una vez finalizado la instalación y configuración de nuestro entorno de trabajo, se procede a realizar la documentación correspondiente a levantamiento de información por medio de Historias de usuarios, escenarios de pruebas y elebaroacion de scripts.

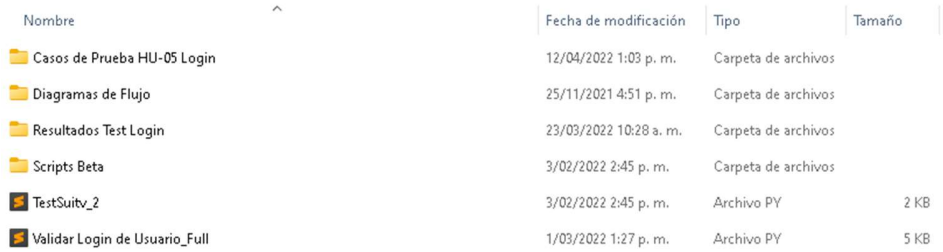
8.1 Diseño de Casos de Pruebas por Funcionalidades

Para la funcionalidad correspondiente a Login de usuarios, se desarrollo por parte del Product Owner del proyecto la HU-005_Login, en la cual se evidencian 06 criterios de aceptación y precondiciones o requisitos dispuestos para que el desarrollador de la funcionalidad tuviera en cuenta en el momento de abordarla, posterior al análisis de la historia de usuario, se procede a realizar el documento correspondiente a la elaboración de casos de pruebas, la documentación elaborada para el desarrollos de script, será entregada a la empresa en un repositorio indicado por ellos mismos, teniendo en cuenta que es información confidencial, no es posible visualizarla detallada mente en este documento.



Nombre	Fecha de modificación	Tipo	Tamaño
24-11-2021 Casos de Prueba Login	1/12/2021 12:06 p. m.	Hoja de cálculo d...	75 KB
Anexo1_HU-005	2/07/2021 1:21 p. m.	Documento de Mi...	35 KB
HU-005_Login	7/09/2021 1:23 p. m.	Documento de Mi...	13 KB
Proceso	21/01/2022 9:32 a. m.	Archivo PNG	54 KB

Ilustración 7 Documentación Historia de usuarios y criterios de aceptación Automatizados-Fuente Propia



Nombre	Fecha de modificación	Tipo	Tamaño
Casos de Prueba HU-05 Login	12/04/2022 1:03 p. m.	Carpeta de archivos	
Diagramas de Flujo	25/11/2021 4:51 p. m.	Carpeta de archivos	
Resultados Test Login	23/03/2022 10:28 a. m.	Carpeta de archivos	
Scripts Beta	3/02/2022 2:45 p. m.	Carpeta de archivos	
TestSuitv_2	3/02/2022 2:45 p. m.	Archivo PY	2 KB
Validar Login de Usuario_Full	1/03/2022 1:27 p. m.	Archivo PY	5 KB

Ilustración 8 Documentación General Historia de usuarios y criterios de aceptación Automatizados-Fuente Propia

8.2 Scripts Desarrollados por Escenarios de Pruebas

La estructura que tiene actualmente el script desarrollado para lograr la correcta automatización de los escenarios de pruebas para el proceso de Login de usuario es la siguiente:

AREANDINA
Fundación Universitaria del Área Andina

Línea(s) de código	Significado
import HtmlTestRunner import unittest from selenium import webdriver from HtmlTestRunner import HTMLTestRunner from selenium.webdriver.common.by import By from selenium.webdriver.common.keys import Keys from selenium.webdriver.support import expected_conditions as EC import time	el primer paso es la Importacion de librerias de selenium y python con el fin de hacer uso de las clases, métodos y atributos que las conforman.
from selenium.webdriver.support.wait import WebDriverWait	El segundo paso es crear una instancia de Chrome que luego se puede controlar con comandos de Selenium. Hay previsto un tiempo de espera en segundos para que arranque el navegador.

class WorkManagerTest(unittest.TestCase):
 	
def setUp(self):
self.driver = webdriver.Chrome(executable_path=r"P:\Curso Selenium\chromedriver")
	El script llamará la ruta en la cual se encuentra alojado nuestro driver para ejecucion del script en el navegador Chrome de forma automatica

class WorkManagerTest(unittest.TestCase):
 	
def setUp(self):
 	
self.driver.get('http://dev.workmanagercloud.com/')
	El script llamará la pagina http://dev.workmanagercloud.com, en la cual se realizaran las automatizaciones o validaciones de los creiterios de aceptacion.

search_field.send_keys("Término de búsqueda")search_field.submit()	A continuación, se introduce el texto del término de búsqueda con el método submit().
ldef test_01_search(self): pass def test_02_validate_form_login(self):	Estas lineas de codigo , corresponden a las validaciones de prueba que se ejecutaran, se nombran con el nombre del test que se realizara la ejecucion
def tearDown(self): self.driver.close()	El código indica a la instancia del navegador que debe cerrarse.
if __name__ == '__main__': unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(output="Resultados Test Login"))	La línea final del código indica al script la creacion del reporte correspondiente a los resultados de la automatizacion .

NOTA: es necesario tener en cuenta que la complejidad o la cantidad de los test, varían de acuerdo a los criterios de aceptación tenidos en cuenta por la historia de usuarios y los escenarios de pruebas indicados en la documentación para ser desarrollados y ejecutados.

Ilustración 9 Descripción general de la estructura del script de automatización

```
import HtmlTestRunner
import unittest
from selenium import webdriver
from HtmlTestRunner import HTMLTestRunner
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions as EC
import time

from selenium.webdriver.support.wait import WebDriverWait


class WorkManagerTest(unittest.TestCase):


    def setUp(self):
 #Se indica al escript la ruta en el driver para la ejecucion del navegador de forma automatica
        self.driver = webdriver.Chrome(executable_path=r"P:\Curso Selenium\chromedriver")

 # Se indica al escript la ruta en el navegador que debe abrir
        self.driver.get('http://dev.workmanagercloud.com/')

 # Inicio de los test a nivel de criterios de aceptacion....

    def test_01_search(self):
        pass


    def test_02_validate_form_login(self):

# se indica al script  el cierre del navegador

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

 # codigo para generacion de reporte por medio de pyunitreport
if __name__ == '__main__':
    unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(output="Resultados Test Login"))
```

Ilustración 10 Arquitectura de Código Script automatizaciones-Fuente Propia

Teniendo en cuenta el orden de los escenarios de pruebas identificados en el documento de pruebas, se realiza la automatización de los siguientes escenarios:

def test_01_search
def test_02_validate_form_login
def test_03_validate_field_password
def test_04_validate_field_username
def test_05_login_success

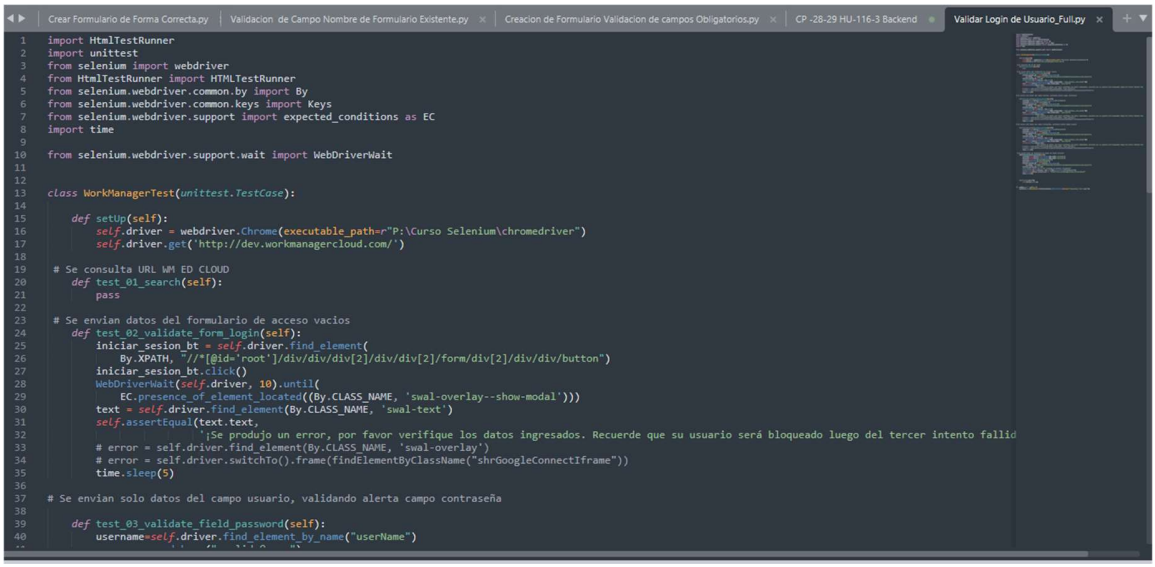


Ilustración 11 Script de Automatización Login de Usuario-Fuente Propia

La anterior imagen muestra el script desarrollado para realizar la automatización del proceso de Login, teniendo en cuenta los escenarios de pruebas expuestos en la documentación desarrollada.

8.3 Resultado Esperado

9. Posterior al desarrollo del código correspondiente a cada criterio de aceptación, se debe realizar ejecución del mismo, con el fin de validar en los resultados del test, que cada criterio o ejecución de prueba de forma automatizada, estén acorde a lo estipulado en cada línea de código, es preciso indicar que quien interactúe con el script desarrollado debe contar con algún conocimiento en programación, adicional a esto entender la documentación a nivel de Historia de usuario y documento de análisis y diseño de casos de pruebas, esto con el fin pueda entender lo que aquí se realiza, por otra parte estas ejecuciones automáticas , ahorraran tiempo en pruebas de regresión sobre la funcionalidad.

```
mport HtmlTestRunner
mport unittest
rom selenium import webdriver
rom HtmlTestRunner import HTMLTestRunner
rom selenium.webdriver.common.by import By
rom selenium.webdriver.common.keys import Keys
rom selenium.webdriver.support import expected_conditions as EC
mport time

rom selenium.webdriver.support.wait import WebDriverWait


lass WorkManagerTest(unittest.TestCase):

   def setUp(self):
       self.driver =s=Service=webdriver.Chrome("P:\Curso Selenium\chromedriver")
       self.driver.get('http://dev.workmanagercloud.com/')

# Se consulta URL WM ED CLOUD
```

```
Running tests...
----------------------------------------------------------------------
 test_01_search (__main__.WorkManagerTest) ... OK (6.123612)s
 test_02_validate_form_login (__main__.WorkManagerTest) ... OK (11.315256)s
 test_03_validate_field_password (__main__.WorkManagerTest) ... OK (8.573940)s
 test_04_validate_field_username (__main__.WorkManagerTest) ... OK (8.667697)s
 test_05_login_success (__main__.WorkManagerTest) ... OK (10.911549)s

----------------------------------------------------------------------
Ran 5 tests in 0:00:55

OK



Generating HTML reports...
Resultados Test Login\TestResults___main__.WorkManagerTest_2022-04-12_13-46-42.html
[Finished in 56.2s]
```

Ilustración 12 Ejemplo de Resultados de ejecución de script-Fuente Propia

10. Actividad 2 Automatización de Funcionalidad Creación de Formularios

Para la funcionalidad correspondiente a Creación de Formularios, se desarrolló por parte del Product Owner del proyecto la HU-054, en la cual se evidencian 04 criterios de aceptación y precondiciones o requisitos dispuestos para que el desarrollador de la funcionalidad tuviera en cuenta en el momento de abordarla, posterior al análisis de la historia de usuario, se procede a realizar el documento correspondiente a la elaboración de casos de pruebas, la documentación elaborada para el desarrollos de script, será entregada a la empresa en un repositorio indicado por ellos mismos, teniendo en cuenta que es información confidencial, no es posible visualizarla detallada mente en este documento, adicionalmente se aborda HU-081 Creación de Campos en Formularios , se realiza procedimiento de elaboración de escenarios de pruebas y posterior mente automatización de los mismos.

10.1 Scripts Desarrollados por Escenarios de Pruebas

Para la automatización de los escanearlos de pruebas para las HU-054, se desarrollaron 12 escenarios de pruebas los cuales fueron automatizados en su totalidad, abarcando escenarios de pruebas tales como:

- Validaciones de campos para creación de formularios

- Validación de caracteres especiales
- Validación de listas
- Validación de alertas

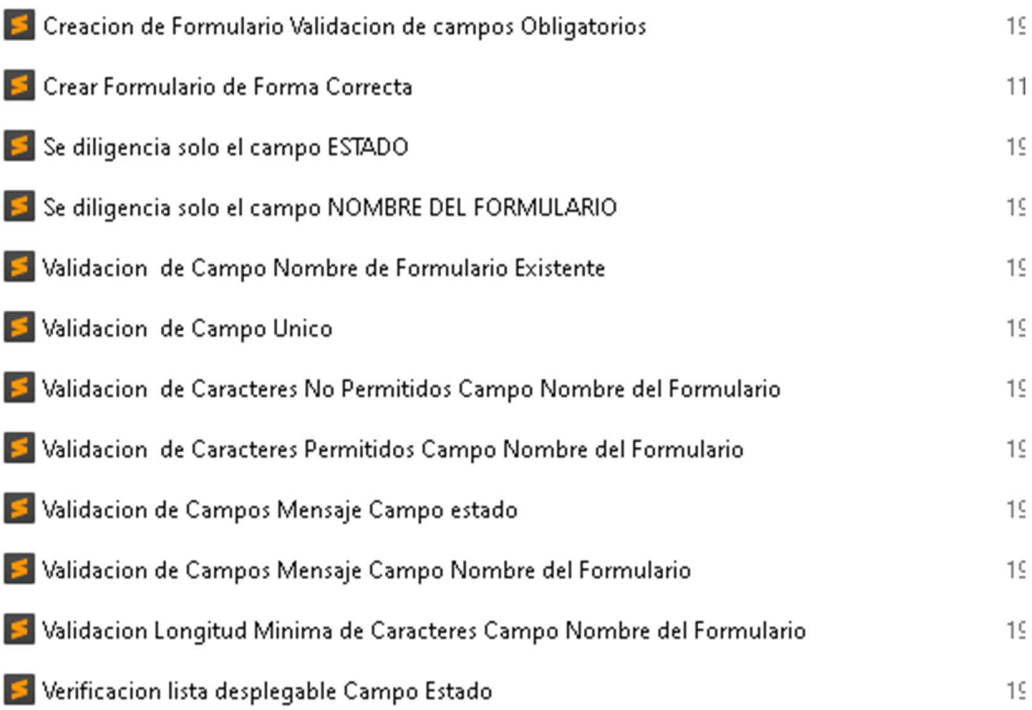


Ilustración 13 Script Automatizados Creación de Formularios Fuente Propia

En el desarrollo del script para la creación de formularios, se contempla la estructura mencionada anteriormente para la validación de Login, se trabaja en la misma estructura de desarrollo con Selenium.

Crear Formulario de Forma Correcta.py

```python
import HtmlTestRunner
import unittest
from selenium import webdriver
from HtmlTestRunner import HTMLTestRunner
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
from selenium.webdriver.support import expected_conditions as EC
import time

from selenium.webdriver.support.wait import WebDriverWait


class WorkManagerTest(unittest.TestCase):


    def setUp(self):
        self.driver =s=Service=webdriver.Chrome("P:\Curso Selenium\chromedriver")
        self.driver.get('https://dev.workmanagercloud.com')


 # Crear Formulario de Forma Correcta

    def test_01_create_form_success(self):
        username=self.driver.find_element(By.NAME,"userName")
        password=self.driver.find_element(By.NAME,"password")
        username.send_keys("***")
        password.send_keys("***")
        iniciar_sesion_bt = self.driver.find_element(
            By.XPATH, "//*[@id='root']/div/div/div[2]/div/div[2]/form/div[2]/div/div/button")
        iniciar_sesion_bt.click()
# Se espera hasta que la URL contenga la palabra 'dashboard'
        WebDriverWait(self.driver, 10).until(EC.url_contains('dashboard'))
        assert self.driver.current_url == 'https://dev.workmanagercloud.com/dashboard'
# Seleccionar opcion Crear Formulario
        WebDriverWait(self.driver, 20)\
        .until(EC.element_to_be_clickable((By.XPATH,
            '/html/body/div/div/div[3]/div[3]/div[1]/div/div[2]/div/div/div/div/div[2]/div[3]/ul/ul/li[3]/a/span')))\
        .click()
        time.sleep(3)
#Diligencimaiento campos del formulario
        nombre_del_formulario=self.driver.find_element(By.NAME,"name")
        descripcion=self.driver.find_element(By.CSS_SELECTOR,"textarea")
        estado=self.driver.find_element(By.XPATH, "//*[@id='status']")
```

Ilustración 14 Automatización Creación de Campos de Forma Correcta

Nota: como se evidencia en la imagen anterior, cada paso de la automatización es comenta riada, esto con el fin de ser de fácil entendimiento de quien pueda ejecutar las pruebas, por otro lado, se debe tener en cuenta que existen datos únicos los cuales deben ser remplazados en caso de correr el script, como por ejemplo datos de usuario de acceso, nombres ya existentes de formularios etc.

Para las automatizaciones de la HU-081 correspondiente a la creación de campos en formularios, se toma en cuenta dos escenarios:

1- Creación de Campos por medio de la Creación de Formularios
2- Creación de Campos Por medio de la Consulta sobre Formularios

10.2 Creación de Capos por medio de la Creación de Formularios

Para la creación de campos, realizando el proceso de crear formulario y seguido los campos sobre el formulario, se desarrollaron 10 script, en los cuales se logra la automatización de:

- Botones de continuidad de procesos
- Validación de Tipos de Datos
- Validación de listas desplegables
- Validación de Longitud de caracteres

- Crear Validacion Dato Campo Longitud máxima y Minima de Forma Correcta
- Crear Form Validacion Dato Campo Longitud máxima de caracteres
- Crear Form Seleccion Dato tipo Cadena de Texto Control Caja de texto
- Crear Form Validacion Seleccion de Dato_Accion Boton Siguiente
- Crear Form Validacion Boton Siguiente
- Crear Form Validacion Seleccion Tipo Control
- Crear Form Validacion tipo de dato Fecha
- Crear Form Validacion tipo de dato Entero
- Crear Form Validacion tipo de dato Cadena de texto
- Crear Form Visualizacion Lista Desplegable Con Tipo de Dato

Ilustración 15 Script desarrollador para el proceso de creación de campos sobre formularios

Nota: En el desarrollo del script para la creación de campos de formularios, se contempla la estructura mencionada anteriormente para la validación de Login y la creación de formularios, se trabaja en la misma estructura de desarrollo con Selenium, contrariando líneas de código para fácil entendimiento de quien ejecuta y exportando las librerías de Selenium.

Crear Form Validacion tipo de dato Entero.py

```
            '/html/body/div/div/div[3]/div[3]/div[1]/div/div[2]/div/div/div/div/div[2]/div[3]/ul/ul/li[3]/a/span')))\
        .click()
        time.sleep(3)
#Diligencimaiento campos del formulario
        nombre_del_formulario=self.driver.find_element(By.NAME,"name")
        descripcion=self.driver.find_element(By.CSS_SELECTOR,"textarea")
        estado=self.driver.find_element(By.XPATH, "//*[@id='status']")
        nombre_del_formulario.send_keys("Pasantias ED Cloud v18")
        time.sleep(3)
        descripcion.send_keys("prueba")
        time.sleep(3)
# Verificacion opciones de la lista
        opcion=estado.find_elements(By.TAG_NAME,"option")
        for option in opcion:
            print("Los valores son: %s" % option.get_attribute("value"))
            option.click()
            time.sleep(2)
# seleccionar estado de la lisda con ID 2 (activo)
        seleccionar_estado=Select(self.driver.find_element(By.XPATH, "//*[@id='status']"))
        seleccionar_estado.select_by_value("2")
        time.sleep(2)
# Dar clic en boton Crear
        crear_sesion_bt = self.driver.find_element(By.XPATH, "//*[@id='root']/div/div[3]/div[3]/div[1]/div/div[2]/div/div/div/form/div[4]/button")
        crear_sesion_bt.click()
        time.sleep(3)
# Validar mensaje creacion de Campos
        WebDriverWait(self.driver, 10).until(EC.presence_of_element_located((By.CLASS_NAME, 'swal-overlay')))
        text = self.driver.find_element(By.CLASS_NAME, 'swal-text')
        self.assertEqual(text.text,'¡Por favor cree los campos para el formulario!')
        time.sleep(3)
# Dar clic en boton Crear campo
        crear_campo_bt = self.driver.find_element(By.XPATH, "//*[@id='panel-0']/div/div/div/div[1]/div/div[2]/div/div/button")
        crear_campo_bt.click()
        time.sleep(3)
# Dar Seleccionar Dato Entero
        Select(self.driver.find_element_by_id("type_data")).select_by_visible_text("Entero")
        time.sleep(3)
# Visualizacion de Control de Datos
        self.driver.find_element_by_id("control_data").click()
        time.sleep(3)
        Select(self.driver.find_element_by_id("control_data")).select_by_visible_text(u"Caja de texto numérica")
        time.sleep(3)
```

Ilustración 16 Validación de Capos desde la Creación de Formularios Fuente propia

10.3 Creación de Campos por medio de la Consulta de Formularios

Para la creación de campos, realizando el proceso de consultar un formulario y seguido los campos sobre el formulario, se toma como parámetros de búsqueda y de pruebas el formulario ya creado en la aplicación WorkManager ED Cloud ® , para ello se utiliza el **Xpath** del formulario deseado , con el fin de que la búsqueda del mismo sea exacta, teniendo en cuenta lo anterior, se desarrollan 11 script, en los cuales se logra la automatización de:

- Botones de continuidad de procesos
- Validación de Tipos de Datos
- Validación de listas desplegables
- Validación de Longitud de caracteres

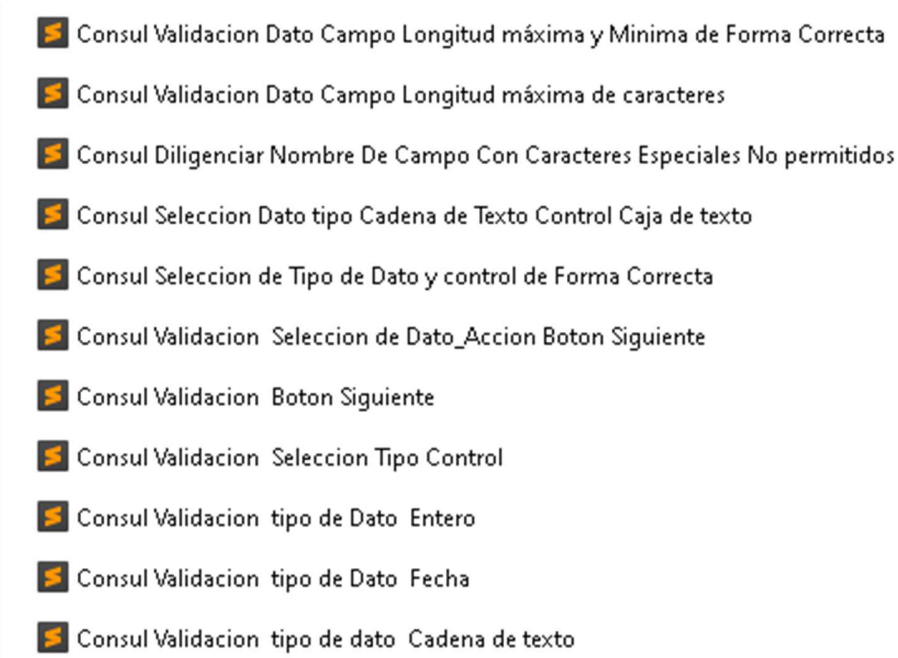
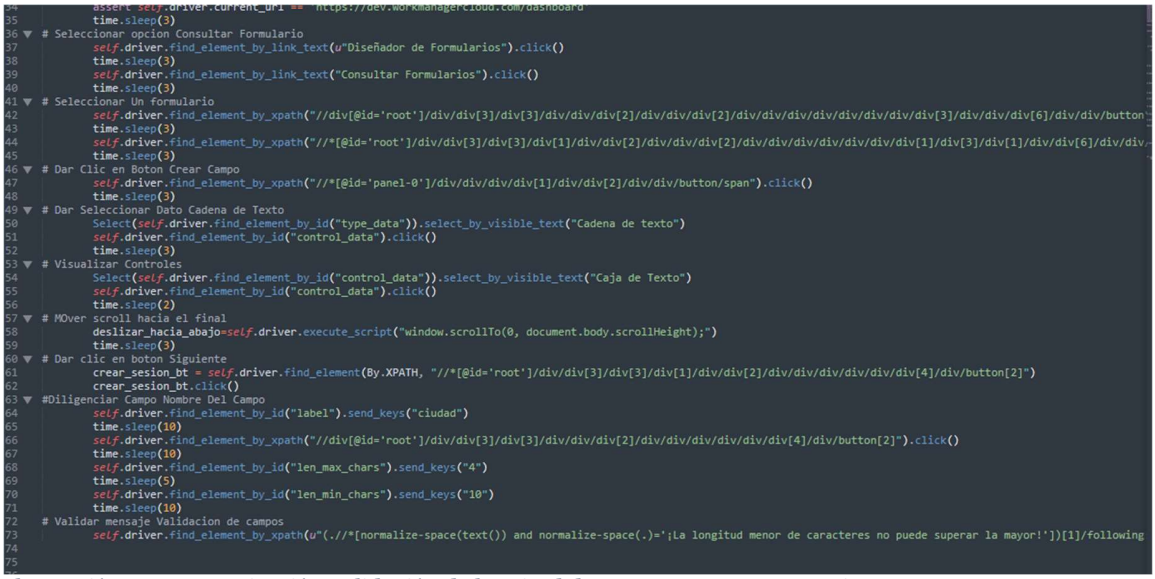


Ilustración 17Script desarrollador para el proceso de creación de campos sobre la consulta de formularios

Nota: En el desarrollo del script para la creación de campos de formularios, se contempla la estructura mencionada anteriormente para la validación de Login y la creación de formularios, se trabaja en la misma estructura de desarrollo con Selenium, contrariando líneas de código para fácil entendimiento de quien ejecuta y exportando las librerías de Selenium.



```
        assert self.driver.current_url == "https://dev.workmanagercloud.com/dashboard"
        time.sleep(3)
# Seleccionar opcion Consultar Formulario
        self.driver.find_element_by_link_text(u"Diseñador de Formularios").click()
        time.sleep(3)
        self.driver.find_element_by_link_text("Consultar Formularios").click()
        time.sleep(3)
# Seleccionar Un formulario
        self.driver.find_element_by_xpath("//div[@id='root']/div/div[3]/div[3]/div/div/div[2]/div/div/div[2]/div/div/div/div/div/div/div[3]/div/div/div[6]/div/div/button
        time.sleep(3)
        self.driver.find_element_by_xpath("//*[@id='root']/div/div[3]/div[3]/div[1]/div/div[2]/div/div/div[2]/div/div/div/div/div/div/div[1]/div[3]/div[1]/div/div[6]/div/div
        time.sleep(3)
# Dar Clic en Boton Crear Campo
        self.driver.find_element_by_xpath("//*[@id='panel-0']/div/div/div/div[1]/div/div[2]/div/div/button/span").click()
        time.sleep(3)
# Dar Seleccionar Dato Cadena de Texto
        Select(self.driver.find_element_by_id("type_data")).select_by_visible_text("Cadena de texto")
        self.driver.find_element_by_id("control_data").click()
        time.sleep(3)
# Visualizar Controles
        Select(self.driver.find_element_by_id("control_data")).select_by_visible_text("Caja de Texto")
        self.driver.find_element_by_id("control_data").click()
        time.sleep(2)
# MOver scroll hacia el final
        deslizar_hacia_abajo=self.driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(3)
# Dar clic en boton Siguiente
        crear_sesion_bt = self.driver.find_element(By.XPATH, "//*[@id='root']/div/div[3]/div[3]/div[1]/div/div[2]/div/div/div/div/div/div[4]/div/button[2]")
        crear_sesion_bt.click()
#Diligenciar Campo Nombre Del Campo
        self.driver.find_element_by_id("label").send_keys("ciudad")
        time.sleep(10)
        self.driver.find_element_by_xpath("//div[@id='root']/div/div[3]/div[3]/div/div/div[2]/div/div/div/div/div/div/div[4]/div/button[2]").click()
        time.sleep(10)
        self.driver.find_element_by_id("len_max_chars").send_keys("4")
        time.sleep(5)
        self.driver.find_element_by_id("len_min_chars").send_keys("10")
        time.sleep(10)
# Validar mensaje Validacion de campos
        self.driver.find_element_by_xpath(u".//*[normalize-space(text()) and normalize-space(.)='¡La longitud menor de caracteres no puede superar la mayor!'])[1]/following
```

Ilustración 18 Automatización validación de longitud de caracteres Fuente Propia

10.4 Resultado esperado

11. Posterior al desarrollo del código correspondiente a cada criterio de aceptación, se debe realizar ejecución del mismo, con el fin de validar en los resultados del test, que cada criterio o ejecución de prueba de forma automatizada, estén acorde a lo estipulado en cada línea de código, es preciso indicar que quien interactúe con el script desarrollado debe contar con algún conocimiento en programación, adicional a esto entender la documentación a nivel de Historia de usuario y documento de análisis y diseño de casos de pruebas, esto con el fin pueda entender lo que aquí se realiza, por otra parte estas ejecuciones automáticas , ahorraran tiempo en pruebas de regresión sobre la funcionalidad.

12. Actividad 4 Automatización de Funcionalidad Radicación de Formularios

Con relación al proceso de automatización, correspondiente a la funcionalidad de Radicación de formularios, se aborda la Historia de usuario HU-057, en la cual se definen aquellos criterios de aceptación tenidos en cuenta para la realización del desarrollo a nivel de Backend y frontend y posterior mente y con el objetivo de esta pasantía, realizar la automatización de las pruebas.

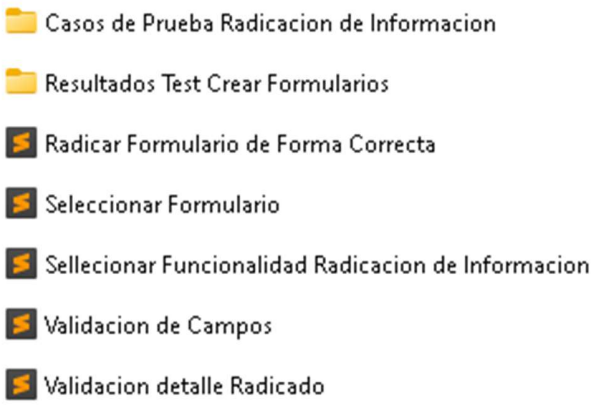
Teniendo en cuenta lo anteriormente descrito, se realiza de la misma manera que las funcionalidades automatizadas anteriormente, el análisis y desarrollo de casos de pruebas, en esta funcionalidad se debe tener en cuenta que no todos los formularios desarrollados por un usuario o varios usuarios contienen la misma estructura, por tal razón para el proceso de radicación, se validara criterios tales como la existencia de la funcionalidad, la visualización de formularios, en vio de

proceso de radicación, entre otros criterios de aceptación relacionadas a la Historia de usuarios.

12.1 Scripts Desarrollados por Escenarios de Pruebas

Para el proceso de automatización de radicación, se continúa manejando la misma estructura de desarrollo, la cual es la importación de librerías, creación de etiquetas de pruebas, comentario de líneas o acciones dentro del script, para esta funcionalidad y teniendo en cuenta que su nivel de complejidad de acuerdo a la historia de usuario desarrollada, se crearon 5 scripts los cuales abordan los 4 criterios de aceptación planteados en la HU, se desarrollan de acuerdo a la existencia de:

- Existencia de la funcionalidad de radicación
- Botones para realizar procesos de radicación
- Validación dentro de la radicación
- Detalle del proceso o radicado.



Como se indicaba anterior mente, la estructura del script contempla librerías que Selenium requiere para la correcta ejecución con relación al lenguaje de programación Python, así como comentario de líneas de código, para mayor entendimiento de quien realice la ejecución del script.

```python
import HtmlTestRunner
import unittest
from selenium import webdriver
from HtmlTestRunner import HTMLTestRunner
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
from selenium.webdriver.support import expected_conditions as EC
import time

from selenium.webdriver.support.wait import WebDriverWait


class WorkManagerTest(unittest.TestCase):


    def setUp(self):
        self.driver =s=Service=webdriver.Chrome("P:\Curso Selenium\chromedriver")
        self.driver.get('https://dev.workmanagercloud.com')


 # Crear Formulario de Forma Correcta

    def test_01_create_form_success(self):
        username=self.driver.find_element(By.NAME,"userName")
        password=self.driver.find_element(By.NAME,"password")
        username.send_keys("**********")
        password.send_keys("**********")
        iniciar_sesion_bt = self.driver.find_element(
            By.XPATH, "//*[@id='root']/div/div/div[2]/div/div[2]/form/div[2]/div/div/button")
        iniciar_sesion_bt.click()
# Se espera hasta que la URL contenga la palabra 'dashboard'
        WebDriverWait(self.driver, 10).until(EC.url_contains('dashboard'))
        assert self.driver.current_url == 'https://dev.workmanagercloud.com/dashboard'
        time.sleep(5)
# Seleccionar opcion Radicacion Informacion
        WebDriverWait(self.driver, 20)\
        .until(EC.element_to_be_clickable((By.XPATH,
            "//*[@id='root']/div/div[3]/div[2]/div[4]/div[1]/div[2]/ul/li/a ")))\
        .click()
        time.sleep(3)


    def tearDown(self):
```

Ilustración 19 Automatización Existencia Funcionalidad de Radicacion_Fuente Propia

Nota: Cada script desarrollado cuenta con la misma estructura de código y lenguaje de programación.

12.2 Resultado Esperado

Posterior al desarrollo del código correspondiente a cada criterio de aceptación, se debe realizar ejecución del mismo, con el fin de validar en los resultados del test, que cada criterio o ejecución de prueba de forma automatizada, estén acorde a lo estipulado en cada línea de código, es preciso indicar que quien interactúe con el script desarrollado debe contar con algún conocimiento en programación, adicional a esto entender la documentación a nivel de Historia de usuario y documento de análisis y diseño de casos de pruebas, esto con el fin pueda entender lo que aquí se realiza, por otra parte estas ejecuciones automáticas , ahorraran tiempo en pruebas de regresión sobre la funcionalidad.

13. Conclusiones

La etapa de pasantías, ha representado en mi desarrollo intelectual y personal un fundamento indispensable para el proceso de aprendizaje, esto dado a que se me ha permitido aumentar por medio de la experiencia, conocer los diferentes contextos de la empresa Estrategias Documentales S.AS, los cuales se deben analizar y tener en cuenta en los procesos de desarrollo de software, lenguajes de programación y procesos de automatizaciones a nivel de QA para la ejecución de pruebas funcionales.

Las actividades anteriormente descritas, las funcionalidades automatizadas, de acuerdo al plan de trabajo y propuestas de automatización aprobadas, han sido cumplidas en su totalidad y de manera satisfactoria, por ello se puede enfatizar que el proceso de pasantías ha sido provechoso al máximo para todos los involucrados, por ejemplo y hablando personalmente , como alumno me siento con conocimientos adquiridos en terrenos cuya existencia de vacíos en meses anteriores eran algo grandes, a la fecha y con la finalización de las pasantías se han adquirido conocimientos grandes, haciendo en mi una mano de obra capacitada y de calidad, por otro lado la universidad, quien una ves ha cumplido su visión de forma exitosa, la empresa Estrategias Documentales SAS, al haber obtenido los servicios y aportes dados por el pasante.

14. Recomendaciones

A la Empresa Estrategias Documentales SAS, seguir ofreciendo oportunidades de capacitar la mano de obra de alumnos de las universidades, instituciones educativas del país, por medio del proceso de pasantías, por otro lado recomendar se continue con el proceso de automatización de funcionalidades, esto dado a que según la experiencia en lo trabajado, este proceso genera pruebas de regresión lo cual en caso de existir fallas en procesos se identifican a tiempo o se puede determinar un plan de acción. Otra recomendación y aspecto a tener en cuenta, es que quien ejecute o interactúe con los script desarrollados en este procesos de automatización de funcionalidades, cuente con algún tipo de conocimiento en el procesos de desarrollo, esto teniendo en cuenta que los drivers de automatización que interactúan con Selenium y el navegador, se vuelven obsoletos mes a mes con relación a las actualizaciones que el proveedor de cada navegador despliegue, así como también el lenguaje de programación puede surgir cambios con relación a las actualizaciones de Selenium, por tl razón existe el riesgo que al momento de ejecutarse alguno de los scripts puedan generar error con el paso del tiempo.

A la Universidad, continuar esforzándose cada día mas para ofrecer a los estudiantes una educación de mayor calidad, teniendo en cuenta que el ámbito laboral y de tecnología esta en constante cambios, mantenerse a la vanguardia en cuanto a las nuevas tecnologías que han surgido posterior al tema mundial de la pandemia COVID-19, continuar con la búsqueda de la calidad y cantidad de conocimientos tecnológicos , que ayuden a mejorar o afianzar el tema de soporte y

cultura general en los alumnos de la universidad del área andina.

15. Glosario

- **Selenium:** Conjunto de utilidades que facilita la labor de obtener juegos de pruebas para aplicaciones web. Para ello nos permite grabar, editar y depurar casos de prueba, que podrán ser ejecutados de forma automática e iterativa
- **Python:** Lenguaje de programación multi plataforma de código abierto y gratuito.
- **CrhomeDriver:** Es un archivo .exe que utiliza su interfaz WebDriver para iniciar el navegador Google **Chrome**
- **HU:** Siglas de Historia de Usuario
- **Historia de Usuario:** Documento en el cual se levanta información correspondiente a un requerimiento o solicitud de mejora, en el cual se estable aquellos criterios de aceptación tenidos en cuenta para la correcta ejecución de la funcionalidad requerida.
- **Script:** secuencia de comendo**.**
- **Librería:** Conjunto de archivos que se utiliza para desarrollar software. Suele estar compuesta de código y datos, y su fin es ser utilizada por otros programas de forma totalmente autónoma

Bibliografía

@ElPSF. (Agosto de 2009). *Fundación de software de Python*. Obtenido de https://www.python.org/doc/versions/

Alvarez, N. (7 de Agosto de 2021). *Curso Python con Selenium* . Obtenido de https://www.youtube.com/playlist?list=PLas30d-GGNa2UW9-1H-NCNrUocvWD9cyh

Juarez, C. L.-G. (20 December 2018). *Automatización de Pruebas con Selenium y Python*.