UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ACG - Auditoría Calidad y Gestión de Sistemas

2024/2025

Francesco Lorenzoni
PCA25403GU

Practica 6 y 7

Selenium

Contents

Chapter 1

Práctica 6 y 7

1.1 Ejercicio 1 - Sample ToDo app

Listing 1.1: SampleTest.py

```
import pytest
from selenium import webdriver
import sys
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from time import sleep

def test_lambdatest_todo_app():

    # Inicializa el WebDriver para Chrome
    chrome_driver = webdriver.Chrome()

    # Navega al URL de la aplicación que queremos testear.
    chrome_driver.get('https://lambdatest.github.io/sample-todo-app/')

    # Maximiza la ventana del navegador para asegurar que todos los
    # elementos de la página sean visibles.
    chrome_driver.maximize_window()

    # Los siguientes dos comandos encuentran elementos en la página web
    # con el metodo find_element. Los ejemplos usan el atributo 'name'
    # para encontrarlos y realizan un click() en ellos.
    # Podemos usar la función de herramienta de inspección del navegador Chrome
    # para inspeccionar los detalles de los elementos web requeridos.

    chrome_driver.find_element(By.NAME,"li1").click()
    chrome_driver.find_element(By.NAME,"li2").click()

    #Verificar el titulo de la pagina
    assert chrome_driver.title == "Sample page - lambdatest.com"

    # Encontrar el textfield para añadir un todo
    todo_text_field = chrome_driver.find_element(By.ID, "sampletodotext")

    # Escri¡bir un texto ahi
    sample_text = "Happy Testing at LambdaTest"
    todo_text_field.send_keys(sample_text)


    # Haz click en el b¡utton add
    chrome_driver.find_element(By.ID,"addbutton").click()

    #ahora tiene que haber un sexto elemento
    chrome_driver.find_element(By.NAME,"li6").click()

    # Extraer y verificar que el texto asociado al 'li6' sea "sample_text"
```

```
    # Asumiendo que el texto está en un <span> justo al lado del input del checkbox
    li6_span_text = chrome_driver.find_element(By.XPATH,
                                               '//input[@name="li6"]/following-sibling::span'
                                               ).text
    assert li6_span_text == sample_text, "El texto asociado al li6 no es el esperado"

    chrome_driver.close()
```

1.2 Ejercicio 2/3/4 - Sauce

Listing 1.2: Sauce.py - Incluye ejercicios 2, 3 y 4

```
from time import sleep
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
import datetime
import pytest

def test_Sauce_title():
    # Inicializa el WebDriver para Chrome
    driver = webdriver.Chrome()

    # Navega al URL de la aplicación que queremos testear.
    driver.get("https://www.saucedemo.com/")

    # Testear si el titulo es correcto
    assert "Swag Labs" in driver.title

    # Cerrar the driver
    driver.quit()

# Ejercicio 2 - Login
def test_Sauce_login():
    # Inicializa el WebDriver para Chrome
    driver = webdriver.Chrome()

    # Navega al URL de la aplicación que queremos testear.
    driver.get("https://www.saucedemo.com/")
    sleep(3)

    # Localizar username y escibir uno
    driver.find_element(By.ID, "user-name").send_keys("standard_user")

    # Localizar password y escibir uno
    driver.find_element(By.ID, "password").send_keys("secret_sauce")

    # Localizar el botón y hacer click
    driver.find_element(By.ID, "login-button").click()
    sleep(5)

    text = driver.find_element(By.CLASS_NAME, "title").text

    # Verificar si el login ha ido correcto, mirando si hemos cambiado de pagina
    assert "products" in text.lower()

    # Cerrar the driver
    driver.quit()

# Ejercicio 3 - Logout
def test_Sauce_logout():
    # Inicializa el WebDriver para Chrome
    driver = webdriver.Chrome()

    # Navega al URL de la aplicación que queremos testear.
    driver.get("https://www.saucedemo.com/")

    #Hacer el login
    driver.find_element(By.ID, "user-name").send_keys("standard_user")
    driver.find_element(By.ID, "password").send_keys("secret_sauce")
    driver.find_element(By.ID, "login-button").click()

    # Localizar el menu, haz click para abrirlo
    driver.find_element(By.ID, "react-burger-menu-btn").click()
    sleep(3)
```

```
    # Localizar logout y hacer click
    driver.find_element(By.ID, "logout_sidebar_link").click()

    # Check we are back at the login page
    inputs = driver.find_elements(By.TAG_NAME, "input")
    assert len(inputs) == 3
    for i in inputs:
        assert (i.get_attribute("name")) in ["user-name", "password", "login-button"]

    # Cerrar the driver
    driver.quit()



# Ejercicio 4 - Añadir producto al carrito y verificar que esta
# test que añade un producto al carito de compras y verifica que el producto esta ahi despues
def test_add_to_cart():
    # Inicializa el WebDriver para Chrome
    driver = webdriver.Chrome()

    # Navega al URL de la aplicación que queremos testear.
    driver.get("https://www.saucedemo.com/")

    # Hacer el login
    driver.find_element(By.ID, "user-name").send_keys("standard_user")
    driver.find_element(By.ID, "password").send_keys("secret_sauce")
    driver.find_element(By.ID, "login-button").click()
    sleep(2)

    # Obtener el nombre del primer producto para verificar después
    product_name = driver.find_element(By.CLASS_NAME, "inventory_item_name").text

    # Añadir el producto al carrito
    driver.find_element(By.CSS_SELECTOR, "button[data-test='add-to-cart-sauce-labs-backpack']"
        ).click()
    sleep(1)

    # Verificar que el contador del carrito muestre 1
    cart_badge = driver.find_element(By.CLASS_NAME, "shopping_cart_badge")
    assert cart_badge.text == "1"

    # Navegar al carrito
    driver.find_element(By.CLASS_NAME, "shopping_cart_link").click()
    sleep(2)

    # Verificar que el producto está en el carrito
    cart_items = driver.find_elements(By.CLASS_NAME, "cart_item")
    assert len(cart_items) == 1

    # Verificar que el producto en el carrito es el mismo que añadimos
    cart_product_name = driver.find_element(By.CLASS_NAME, "inventory_item_name").text
    assert cart_product_name == product_name

    # Cerrar the driver
    driver.quit()
```

1.3 Ejercicio 5/6/7 - PracticeSoftwareTesting

1.3.1 Ejercicio 5 - Formulario de contacto

Listing 1.3: Practica5.py

```
from time import sleep
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
import datetime
import pytest


def test_contact_form_submission():
   # Inicializa el WebDriver para Chrome
   driver = webdriver.Chrome()
   driver.get("https://practicesoftwaretesting.com")
   # Acceder al formulario
   driver.find_element(By.LINK_TEXT, "Contact").click()
   sleep(1)   # Rellenar formulario
   driver.find_element(By.ID, "first_name").send_keys("Alice")
   driver.find_element(By.ID, "last_name").send_keys("Smith")
   driver.find_element(By.ID, "email").send_keys("alice@example.com")
   Select(driver.find_element(By.ID, "subject")).select_by_visible_text("Customer service")
   driver.find_element(By.ID, "message").send_keys(6*"Hola hola!") # 60 caracteres
   # Enviar
   driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
   sleep(1)   # Esperar a que se procese el envío
   # Verificar mensaje de éxito
   success_msg = driver.find_element(By.CLASS_NAME, "alert-success").text
   assert "Thanks for your message" in success_msg
   # Cerrar the driver
   driver.quit()

# Caso positivo 2: Usar diferente valor válido para el dropdown "Subject"
def test_contact_form_different_subject():
   driver = webdriver.Chrome()
   driver.get("https://practicesoftwaretesting.com")
   driver.find_element(By.LINK_TEXT, "Contact").click()
   sleep(1)
   driver.find_element(By.ID, "first_name").send_keys("Bob")
   driver.find_element(By.ID, "last_name").send_keys("Johnson")
   driver.find_element(By.ID, "email").send_keys("bob@example.com")
   Select(driver.find_element(By.ID, "subject")).select_by_visible_text("Webmaster")
   driver.find_element(By.ID, "message").send_keys("Este mensaje tiene más de 50 caracteres
       como es requerido por el formulario de contacto.")

   driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
   sleep(1)   # Esperar a que se procese el envío
   success_msg = driver.find_element(By.CLASS_NAME, "alert-success").text
   assert "Thanks for your message" in success_msg

   driver.quit()

# Caso positivo 3: Usar un mensaje muy largo
def test_contact_form_long_message():
   driver = webdriver.Chrome()
   driver.get("https://practicesoftwaretesting.com")
   driver.find_element(By.LINK_TEXT, "Contact").click()
   sleep(1)
   driver.find_element(By.ID, "first_name").send_keys("Charlie")
   driver.find_element(By.ID, "last_name").send_keys("Brown")
   driver.find_element(By.ID, "email").send_keys("charlie@example.com")
   Select(driver.find_element(By.ID, "subject")).select_by_visible_text("Return")
   driver.find_element(By.ID, "message").send_keys("A" * 200)  # 200 caracteres
```

```
    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)    # Esperar a que se procese el envío
    success_msg = driver.find_element(By.CLASS_NAME, "alert-success").text
    assert "Thanks for your message" in success_msg

    driver.quit()

# Caso positivo 4: Usar caracteres especiales en los campos de nombre
def test_contact_form_special_chars():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com")
    driver.find_element(By.LINK_TEXT, "Contact").click()
    sleep(1)

    driver.find_element(By.ID, "first_name").send_keys("María-José")
    driver.find_element(By.ID, "last_name").send_keys("Rodríguez O'Hara")
    driver.find_element(By.ID, "email").send_keys("maria.jose@example.com")
    Select(driver.find_element(By.ID, "subject")).select_by_visible_text("Payments")
    driver.find_element(By.ID, "message").send_keys("Este mensaje contiene caracteres
        especiales: áéíóú ñ çã. Y tiene más de 50 caracteres.")

    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)    # Esperar a que se procese el envío
    success_msg = driver.find_element(By.CLASS_NAME, "alert-success").text
    assert "Thanks for your message" in success_msg

    driver.quit()

# Caso positivo 5: Usar una dirección de email compleja pero válida
def test_contact_form_complex_email():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com")
    driver.find_element(By.LINK_TEXT, "Contact").click()
    sleep(1)

    driver.find_element(By.ID, "first_name").send_keys("David")
    driver.find_element(By.ID, "last_name").send_keys("Wilson")
    driver.find_element(By.ID, "email").send_keys("david.wilson+test123@sub.example-domain.co.
        uk")
    Select(driver.find_element(By.ID, "subject")).select_by_visible_text("Customer service")
    driver.find_element(By.ID, "message").send_keys("Este es un mensaje de prueba que tiene más
         de cincuenta caracteres como se requiere.")

    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)    # Esperar a que se procese el envío
    success_msg = driver.find_element(By.CLASS_NAME, "alert-success").text
    assert "Thanks for your message" in success_msg

    driver.quit()

# Caso negativo 1: Campo de nombre faltante
def test_contact_form_missing_first_name():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com")
    driver.find_element(By.LINK_TEXT, "Contact").click()
    sleep(1)

    # No se introduce el nombre
    driver.find_element(By.ID, "last_name").send_keys("Smith")
    driver.find_element(By.ID, "email").send_keys("test@example.com")
    Select(driver.find_element(By.ID, "subject")).select_by_visible_text("Customer service")
    driver.find_element(By.ID, "message").send_keys("Este mensaje tiene más de 50 caracteres
        como es requerido por el formulario.")

    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)    # Esperar a que se procese el envío
    # Verificar mensaje de error
    error_msg = driver.find_element(By.CLASS_NAME, "alert-danger").text
    assert "required" in error_msg.lower()
```

```
    driver.quit()

# Caso negativo 2: Email inválido
def test_contact_form_invalid_email():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com")
    driver.find_element(By.LINK_TEXT, "Contact").click()
    sleep(1)

    driver.find_element(By.ID, "first_name").send_keys("John")
    driver.find_element(By.ID, "last_name").send_keys("Doe")
    driver.find_element(By.ID, "email").send_keys("john@") # Email inválido
    Select(driver.find_element(By.ID, "subject")).select_by_visible_text("Customer service")
    driver.find_element(By.ID, "message").send_keys("Este mensaje tiene más de 50 caracteres
        como es requerido por el formulario.")

    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)   # Esperar a que se procese el envío
    # Verificar mensaje de error
    error_msg = driver.find_element(By.CLASS_NAME, "alert-danger").text
    assert "valid email" in error_msg.lower()

    driver.quit()

# Caso negativo 3: Mensaje con menos de 50 caracteres
def test_contact_form_short_message():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com")
    driver.find_element(By.LINK_TEXT, "Contact").click()
    sleep(1)

    driver.find_element(By.ID, "first_name").send_keys("Emma")
    driver.find_element(By.ID, "last_name").send_keys("Garcia")
    driver.find_element(By.ID, "email").send_keys("emma@example.com")
    Select(driver.find_element(By.ID, "subject")).select_by_visible_text("Customer service")
    driver.find_element(By.ID, "message").send_keys("Mensaje corto.") # Menos de 50 caracteres

    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)   # Esperar a que se procese el envío
    # Verificar mensaje de error
    error_msg = driver.find_element(By.CLASS_NAME, "alert-danger").text
    assert "50" in error_msg

    driver.quit()

# Caso negativo 4: Sin selección de asunto
def test_contact_form_no_subject():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com")
    driver.find_element(By.LINK_TEXT, "Contact").click()
    sleep(1)

    driver.find_element(By.ID, "first_name").send_keys("Frank")
    driver.find_element(By.ID, "last_name").send_keys("Miller")
    driver.find_element(By.ID, "email").send_keys("frank@example.com")
    # No se selecciona el asunto
    driver.find_element(By.ID, "message").send_keys("Este mensaje tiene más de 50 caracteres
        como es requerido por el formulario de contacto.")

    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)   # Esperar a que se procese el envío
    # Verificar mensaje de error
    error_msg = driver.find_element(By.CLASS_NAME, "alert-danger").text
    assert "subject" in error_msg.lower()

    driver.quit()

# Caso negativo 5: Apellido faltante
```

```
def test_contact_form_missing_last_name():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com")
    driver.find_element(By.LINK_TEXT, "Contact").click()
    sleep(1)

    driver.find_element(By.ID, "first_name").send_keys("Grace")
    # No se introduce el apellido
    driver.find_element(By.ID, "email").send_keys("grace@example.com")
    Select(driver.find_element(By.ID, "subject")).select_by_visible_text("Customer service")
    driver.find_element(By.ID, "message").send_keys("Este mensaje tiene más de 50 caracteres
        como es requerido por el formulario.")

    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)   # Esperar a que se procese el envío
    # Verificar mensaje de error
    error_msg = driver.find_element(By.CLASS_NAME, "alert-danger").text
    assert "required" in error_msg.lower()

    driver.quit()
```

1.3.2 Ejercicio 6 - Formulario de contacto con adjunto

- ACC2: El nombre del archivo no debe ser editable directamente.
- ACC3: Solo se permiten archivos de tipo `.txt`, `.pdf` o `.jpg`.
- ACC4: El archivo debe tener un tamaño mayor a 0 KB y menor o igual a 500 KB.
- ACC5: Si el archivo no es válido, se debe mostrar un mensaje de error que describa

Figure 1.1: Ejercicio 6 - Criterios de aceptación

Creo que hay un problema con el enunciado, ya que no es posible enviar adjuntos que no son `.txt` vacios. Por este motivo, he ponido solo un caso de prueba positivo, y tres negativos.

Listing 1.4: Practica6.py

```
'''

Ejemplos de casos de prueba:
[Negativo] Se intenta subir un archivo .pdf vacio Aparece un error de tipo
[Negativo] Se intenta subir un archivo .pdf de 100KB  Aparece un error de tamaño
[Negativo] Se intenta subir un archivo .txt de 100KB  Aparece un error de tamaño
[Positivo] Se sube un archivo .txt vacío (0 KB) y se envía correctamente.
Después, automatiza estos casos de prueba utilizando Selenium y pytest.
'''

from time import sleep
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
import datetime
import pytest
import os

# Simulamos las rutas a archivos para pruebas (estos serían archivos reales en un entorno de
    pruebas)
FILE_PDF_120KB = "C:/Users/franc/OneDrive - University of Pisa/Documents/UniPi/UPV/ACG/
    practica67/test_files/sunflower.pdf"
FILE_TXT_EMPTY = "C:/Users/franc/OneDrive - University of Pisa/Documents/UniPi/UPV/ACG/
    practica67/test_files/vacio.txt"
FILE_TXT_100KB = "C:/Users/franc/OneDrive - University of Pisa/Documents/UniPi/UPV/ACG/
    practica67/test_files/lorem.txt"
FILE_PDF_EMPTY = "C:/Users/franc/OneDrive - University of Pisa/Documents/UniPi/UPV/ACG/
    practica67/test_files/vacio.pdf"

def fill_contact_form(driver, file_path=None):
    """Función auxiliar para rellenar formulario de contacto"""
    driver.find_element(By.ID, "first_name").send_keys("Test")
    driver.find_element(By.ID, "last_name").send_keys("User")
    driver.find_element(By.ID, "email").send_keys("test@example.com")
    Select(driver.find_element(By.ID, "subject")).select_by_visible_text("Customer service")
    driver.find_element(By.ID, "message").send_keys(
        "Este mensaje tiene más de 50 caracteres como es requerido por el formulario de
            contacto."
    )

    if file_path:
        # Suponemos que el elemento para subir archivos tiene un ID "file_upload"
        driver.find_element(By.ID, "attachment").send_keys(file_path)

# Caso positivo: Se sube un archivo .txt vacío (0 KB) y se envía correctamente
def test_upload_empty_txt_file():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com")

    # Acceder al formulario
    driver.find_element(By.LINK_TEXT, "Contact").click()
    sleep(1)

    # Rellenar formulario y subir archivo
```

```
    fill_contact_form(driver, FILE_TXT_EMPTY)

    # Enviar el formulario
    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)

    # Verificar mensaje de éxito
    success_msg = driver.find_element(By.CLASS_NAME, "alert-success").text
    assert "Thanks for your message" in success_msg

    driver.quit()

# Caso negativo: Se sube un archivo .pdf de 100kb (no es vacio y el tipo no es permitido)
def test_upload_invalid_file_size_pdf():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com")

    # Acceder al formulario
    driver.find_element(By.LINK_TEXT, "Contact").click()
    sleep(1)

    # Rellenar formulario y subir archivo
    fill_contact_form(driver, FILE_PDF_120KB)

    # Enviar el formulario
    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)

    # Verificar mensaje de error
    error_msg = driver.find_element(By.CLASS_NAME, "alert-danger").text
    assert "file should be empty" in error_msg.lower()

    driver.quit()

# Caso negativo: Se sube un archivo .txt de 100KB (no es vacio)
def test_upload_invalid_file_size_txt():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com")

    # Acceder al formulario
    driver.find_element(By.LINK_TEXT, "Contact").click()
    sleep(1)

    # Rellenar formulario y subir archivo
    fill_contact_form(driver, FILE_TXT_100KB)

    # Enviar el formulario
    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)

    # Verificar mensaje de error
    error_msg = driver.find_element(By.CLASS_NAME, "alert-danger").text
    assert "file should be empty" in error_msg.lower()

    driver.quit()


# Caso negativo: Se sube un archivo .pdf vacio (tipo no permitido)
def test_upload_invalid_file_type_pdf():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com")

    # Acceder al formulario
    driver.find_element(By.LINK_TEXT, "Contact").click()
    sleep(1)

    # Rellenar formulario y subir archivo
    fill_contact_form(driver, FILE_PDF_EMPTY)
```

```
    # Enviar el formulario
    driver.find_element(By.XPATH, '//input[@data-test="contact-submit"]').click()
    sleep(1)

    # Verificar mensaje de error
    error_msg = driver.find_element(By.CLASS_NAME, "alert-danger").text
    assert "file should have a txt extension" in error_msg.lower()

    driver.quit()
```

1.3.3 Ejercicio 7 - Usuario bloqueado tras intentos fallidos de login

Creo que hay un problema aquí también. El .pdf nos no da credenciales de prueba. He registrado un usuario de prueba, que pero no es permanente. Además, si el usuario se bloquea, no es posible desbloquearlo. Mi prueba ha funcionado, la primera vez que la he ejecutada, pero si vuelvo a ejecutarla falla.

Listing 1.5: Practica7.py

```
from time import sleep
from selenium import webdriver
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.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import datetime
import pytest

# Credenciales para pruebas
USERNAME = "kayinik518@idoidraw.com"
CORRECT_PASSWORD = "C4ll!!m3"
INCORRECT_PASSWORD = "wrong_password"

# Función auxiliar para intentar login
def attempt_login(driver, username, password):
    """Intenta iniciar sesión con las credenciales dadas"""
    # Asegurarse de estar en la página de login
    if "auth/login" not in driver.current_url:
        driver.get("https://practicesoftwaretesting.com/auth/login")
        sleep(1)

    # Localizar e introducir username
    driver.find_element(By.ID, "email").clear()
    driver.find_element(By.ID, "email").send_keys(username)

    # Localizar e introducir password
    driver.find_element(By.ID, "password").clear()
    driver.find_element(By.ID, "password").send_keys(password)

    # Localizar el botón y hacer click
    driver.find_element(By.XPATH, '//input[@data-test="login-submit"]').click()
    sleep(1)

# Caso positivo 1: El usuario ingresa correctamente desde el primer intento
def test_successful_login_first_attempt():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com/auth/login")

    # Intentar login con credenciales correctas
    attempt_login(driver, USERNAME, CORRECT_PASSWORD)

    # Verificar que se ha accedido correctamente
    assert "account" in driver.current_url

    driver.quit()

# Caso positivo 2: El usuario ingresa mal la contraseña 2 veces, pero en el tercer intento lo
    hace correctamente
def test_successful_login_after_two_failures():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com/auth/login")

    # Primer intento fallido
    attempt_login(driver, USERNAME, INCORRECT_PASSWORD)

    # Segundo intento fallido
    attempt_login(driver, USERNAME, INCORRECT_PASSWORD)

    # Tercer intento correcto
    attempt_login(driver, USERNAME, CORRECT_PASSWORD)
```

```
    # Verificar que se ha accedido correctamente
    assert "account" in driver.current_url

    driver.quit()

# Caso positivo 3: Usar un usuario diferente después de fallar con otro
def test_successful_login_different_user():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com/auth/login")

    # Tres intentos fallidos con un usuario
    attempt_login(driver, "wrong_user@example.com", INCORRECT_PASSWORD)
    attempt_login(driver, "wrong_user@example.com", INCORRECT_PASSWORD)
    attempt_login(driver, "wrong_user@example.com", INCORRECT_PASSWORD)

    # Intento exitoso con el usuario correcto
    attempt_login(driver, USERNAME, CORRECT_PASSWORD)

    # Verificar que se ha accedido correctamente
    assert "account" in driver.current_url

    driver.quit()

# Caso negativo 1: El usuario falla 3 veces seguidas y queda bloqueado
def test_account_locked_after_three_failures():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com/auth/login")

    # Tres intentos fallidos consecutivos
    attempt_login(driver, USERNAME, INCORRECT_PASSWORD)
    attempt_login(driver, USERNAME, INCORRECT_PASSWORD)
    attempt_login(driver, USERNAME, INCORRECT_PASSWORD)

    # Cuarto intento con contraseña correcta, pero usuario ya bloqueado
    attempt_login(driver, USERNAME, CORRECT_PASSWORD)

    # Verificar mensaje de error que indica que la cuenta está bloqueada
    error_message = driver.find_element(By.CLASS_NAME, "alert-danger").text
    assert "locked" in error_message.lower()

    driver.quit()

# Caso negativo 2: El usuario se equivoca 2 veces, espera unos segundos y falla una tercera
    vez
def test_account_locked_with_delay_between_attempts():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com/auth/login")

    # Dos intentos fallidos consecutivos
    attempt_login(driver, USERNAME, INCORRECT_PASSWORD)
    attempt_login(driver, USERNAME, INCORRECT_PASSWORD)

    # Esperar unos segundos
    sleep(5)

    # Tercer intento fallido
    attempt_login(driver, USERNAME, INCORRECT_PASSWORD)

    # Cuarto intento con contraseña correcta, pero el usuario ya está bloqueado
    attempt_login(driver, USERNAME, CORRECT_PASSWORD)

    # Verificar mensaje de error que indica que la cuenta está bloqueada
    error_message = driver.find_element(By.CLASS_NAME, "alert-danger").text
    assert "locked" in error_message.lower()

    driver.quit()
```

```
# Caso negativo 3: Intentar acceder con un usuario bloqueado (tras realizar 3 intentos
    fallidos)
def test_attempt_login_to_previously_locked_user():
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com/auth/login")

    # Bloquear la cuenta con tres intentos fallidos
    attempt_login(driver, USERNAME, INCORRECT_PASSWORD)
    attempt_login(driver, USERNAME, INCORRECT_PASSWORD)
    attempt_login(driver, USERNAME, INCORRECT_PASSWORD)

    # Cerrar el navegador y abrir uno nuevo (simular nuevo acceso)
    driver.quit()
    driver = webdriver.Chrome()
    driver.get("https://practicesoftwaretesting.com/auth/login")

    # Intentar acceder con el usuario ahora bloqueado
    attempt_login(driver, USERNAME, CORRECT_PASSWORD)

    # Verificar mensaje de error que indica que la cuenta está bloqueada
    error_message = driver.find_element(By.CLASS_NAME, "alert-danger").text
    assert "locked" in error_message.lower()

    driver.quit()
```