

Proyecto QA Integral

1. Información General

Nombre: Alejandro Franco Acosta

Rol: QA Tester

Periodo: Sprint 1 – Sprint 2

Herramientas:

- Jira/Xray
- Python
- Selenium
- Pytest
- JUnit
- XML
- MongoDB
- Mongo Compass
- SoapUI
- Git y GitHub
- Docker
- DOM (Document Object Model)

2. Introducción

El presente informe documenta el trabajo realizado durante el Proyecto QA Integral, cuyo objetivo es demostrar mis conocimientos aplicados al aseguramiento de calidad sobre distintas capas de un sistema, combinando pruebas manuales y automatizadas. El proyecto abarcó validaciones a nivel de API, Base de Datos y UI, manteniendo trazabilidad completa mediante Jira/Xray que es nuestra herramienta de gestión clave para mantener el orden y la planeación.

3. Alcance del Proyecto

Las pruebas realizadas cubren las siguientes áreas:

- API: Validación de servicios SOAP mediante WSDL.
- Base de Datos: Validaciones sobre MongoDB (estructura, consistencia y reglas básicas).
- UI: Automatización del flujo de login en aplicación web.
- Gestión QA: Registro, ejecución y trazabilidad de pruebas en Jira/Xray.

Fuera de alcance:

- Pruebas de performance
- Pruebas de seguridad
- Integración CI/CD

4. Pruebas de API (SOAP)

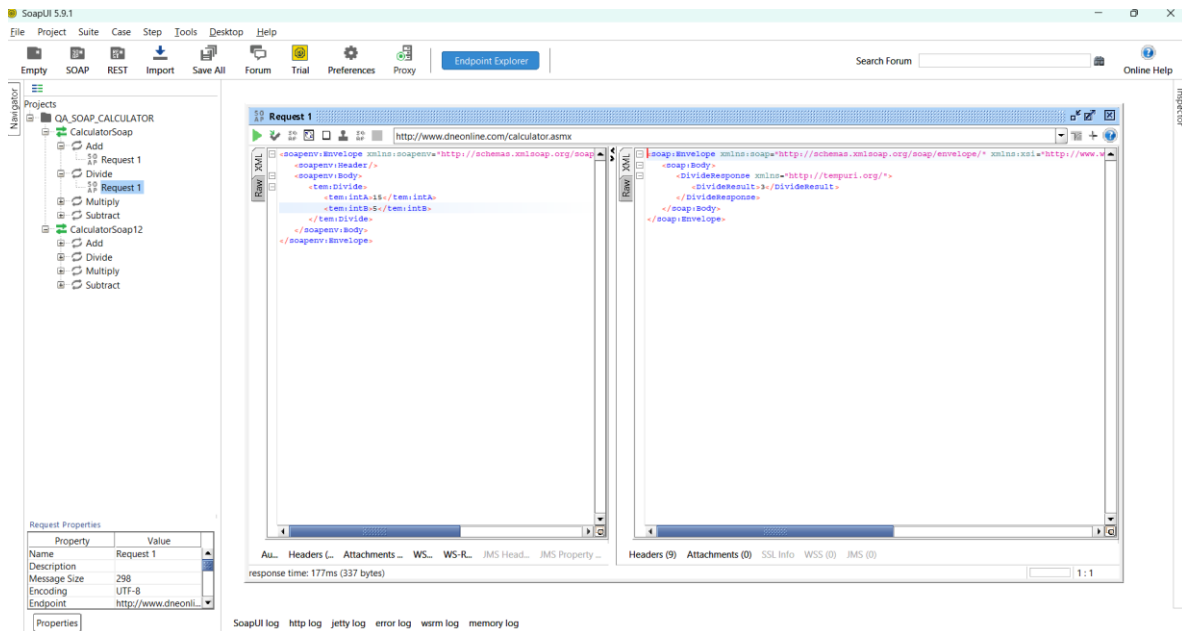
4.1 Descripción

Se validó un servicio SOAP público utilizando SoapUI. A partir del WSDL se generaron operaciones y se crearon Test Suites y Test Cases para validar:

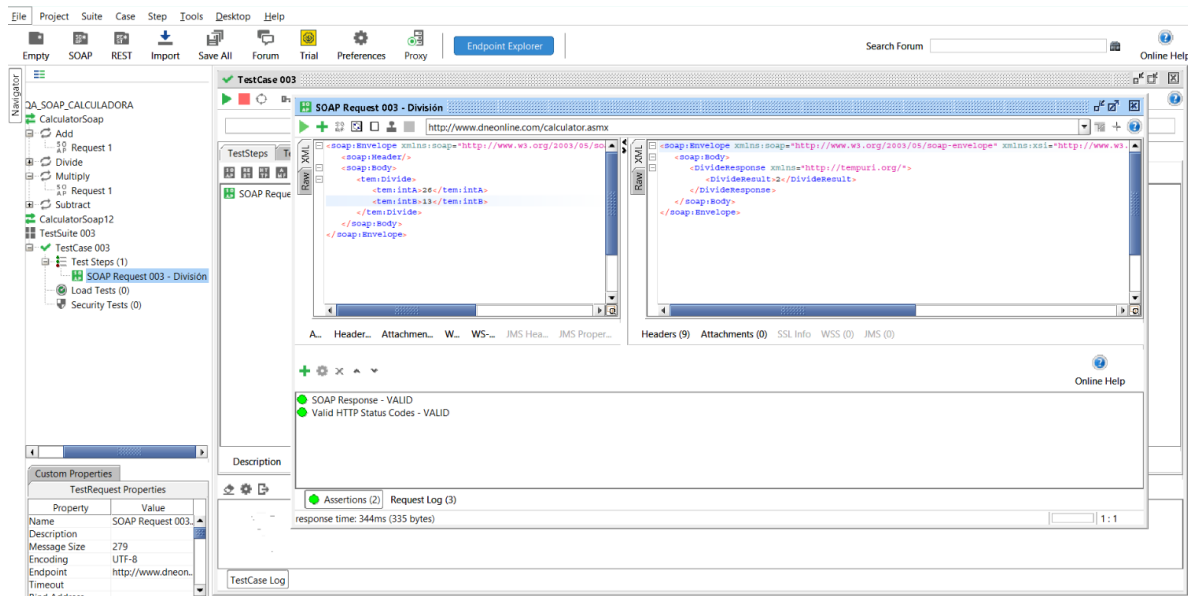
- Disponibilidad del servicio
- Código de respuesta HTTP
- Estructura del mensaje SOAP
- Resultado esperado de la operación

4.2 Evidencias

Request y Response de API en SoapUI (open source)



Test Suite, Test Case y Assertions (contains)



Se registra evidencias en Jira

The screenshot shows a Jira issue page for '1.2 Pruebas de Servicios SOAP'. The issue is assigned to 'alejandro franco' and is in the 'Done' state. The subtasks are listed in a table, all marked as 'DONE'. The details panel on the right shows the assignee, reporter, development actions, labels, due date, start date, sprint, priority, and parent issue.

1.2 Pruebas de Servicios SOAP

Description
Proyecto [GitHub - Alex-af-2024/proyecto-qa-integral](#)

Subtasks

Work	Pri...	Stor...	As...	Status
QA2-9 1.2.1 Importar un WSDL en SoapUI	M		AF	DONE
QA2-10 1.2.2 Crear test suite + test case	M		AF	DONE
QA2-11 1.2.3 Crear assertions(status, XML output, reglas)	M		AF	DONE
QA2-12 1.2.4 Exportar evidencias (XML + capturas)	M		AF	DONE
QA2-13 1.2.5 Registrar caso de prueba manual en Xray	M		AF	DONE

Linked work items
Add linked work item

Details

Assignee: AF alejandro franco

Reporter: AF alejandro franco

Development: Create branch, Create commit

Labels: None

Due date: Dec 16, 2025

Start date: Dec 15, 2025

Sprint: None +1

Priority: Medium

Parent: QA2-11 Proyecto Integral de

4.3 Conocimientos aplicados

- Lectura e interpretación de WSDL
- Estructura de mensajes SOAP
- Uso de assertions funcionales
- Organización de pruebas en suites y casos

4.4 Recomendaciones

- Versionar los proyectos SoapUI en el repositorio
- Separar datos de prueba cuando el servicio lo permita
- Documentar claramente los servicios probados

5. Pruebas de Base de Datos (MongoDB)

5.1 Descripción

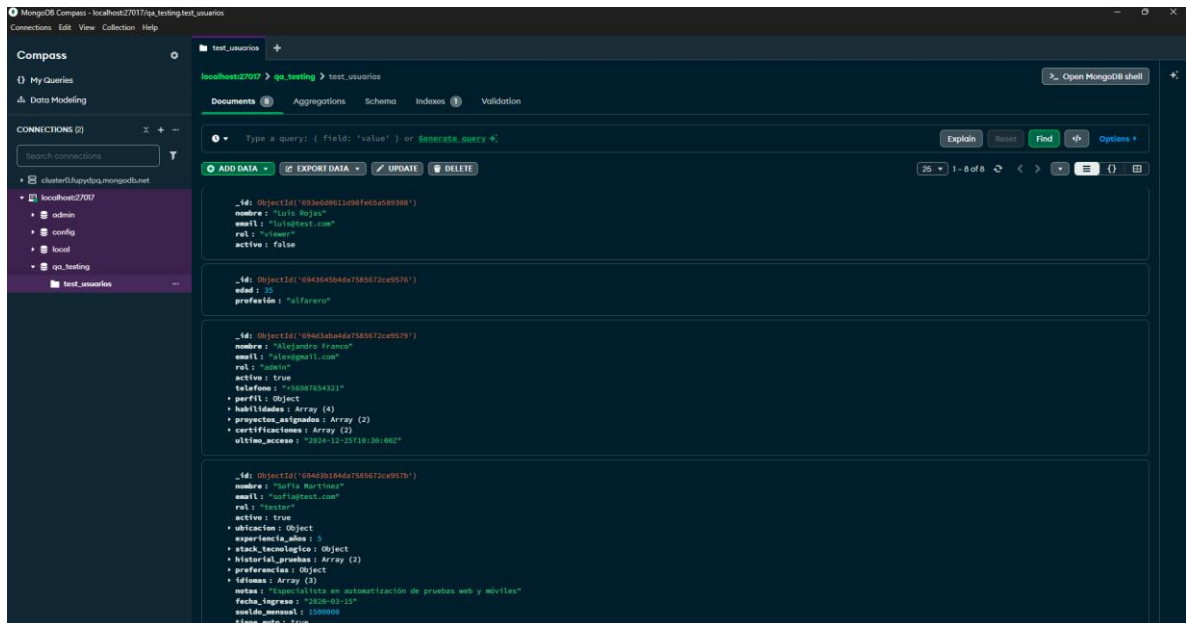
Se crea una base de datos de prueba en MongoDB para validar reglas básicas de integridad y consistencia de datos. Las validaciones se realizaron tanto de forma manual (MongoDB Compass) como automatizada mediante scripts en Python usando PyMongo.

Las validaciones incluyeron:

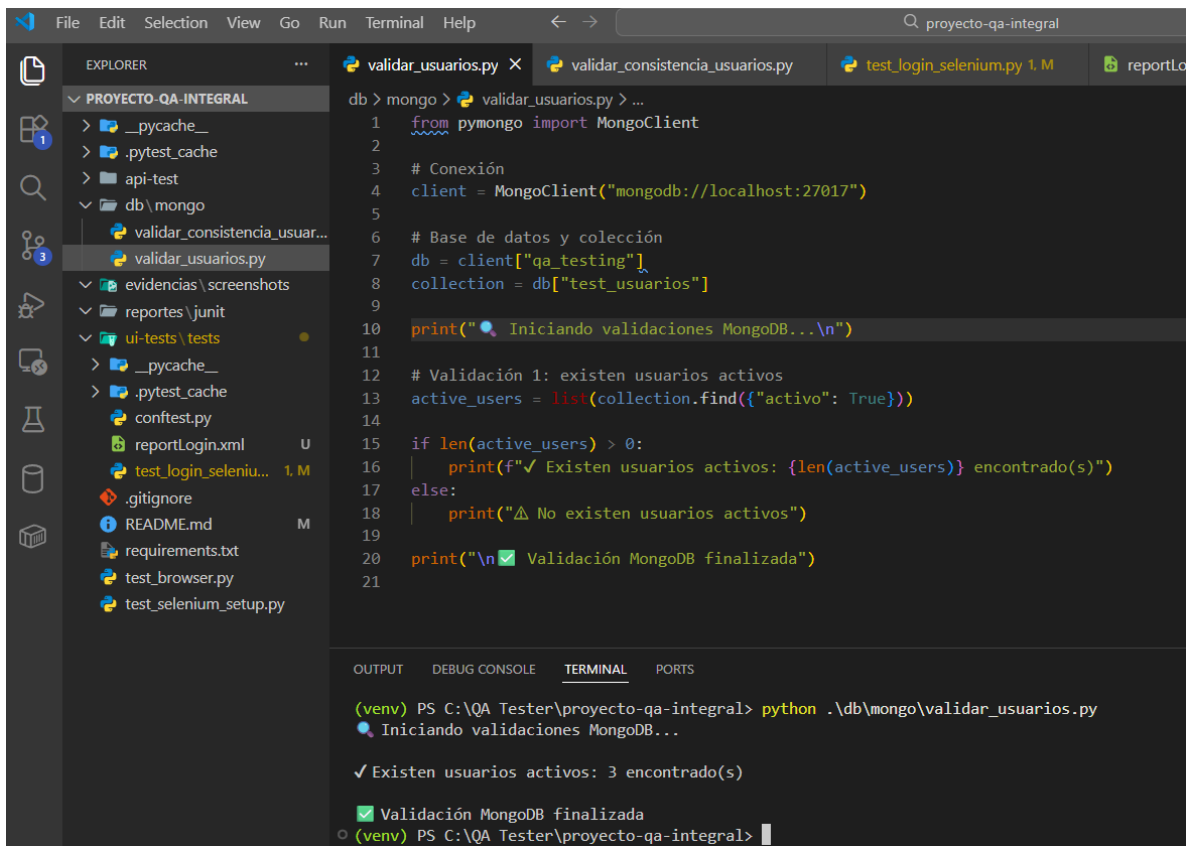
- Existencia de registros
- Validación de usuarios activos
- Validación de formato de emails
- Verificación de campos obligatorios

5.2 Evidencias

Creación de base de datos local con MongoDB que aprovecha el modelado flexible de documentos NoSQL. Visualización en Mongo Compass



Ejecución de script para confirmar la existencia de usuarios activos en DDBB



The screenshot shows a Visual Studio Code editor window with the file explorer on the left and the code editor on the right. The file explorer shows a project named 'PROYECTO-QA-INTEGRAL' with various files and folders. The code editor displays a Python script named 'validar_usuarios.py' which connects to a MongoDB database and checks for active users. The terminal at the bottom shows the command to run the script and its output, indicating that 3 active users were found.

```
db > mongo > python validar_usuarios.py > ...
1 from pymongo import MongoClient
2
3 # Conexión
4 client = MongoClient("mongodb://localhost:27017")
5
6 # Base de datos y colección
7 db = client["qa_testing"]
8 collection = db["test_usuarios"]
9
10 print("🔵 Iniciando validaciones MongoDB...\n")
11
12 # Validación 1: existen usuarios activos
13 active_users = list(collection.find({"activo": True}))
14
15 if len(active_users) > 0:
16     print(f"✓ Existen usuarios activos: {len(active_users)} encontrado(s)")
17 else:
18     print("⚠ No existen usuarios activos")
19
20 print("\n✅ Validación MongoDB finalizada")
21
```

OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(venv) PS C:\QA Tester\proyecto-qa-integral> python .\db\mongo\validar_usuarios.py
🔵 Iniciando validaciones MongoDB...

✓ Existen usuarios activos: 3 encontrado(s)

✅ Validación MongoDB finalizada
(venv) PS C:\QA Tester\proyecto-qa-integral>
```

5.3 Conocimientos aplicados

- Uso de MongoDB Compass
- Modelado flexible de documentos.
- Conexión a MongoDB con PyMongo
- Validaciones lógicas mediante scripts

5.4 Recomendaciones

- Utilizar MongoDB en Docker para mayor portabilidad del proyecto
- Definir validaciones de esquema cuando el proyecto crezca
- Separar claramente datos de prueba y datos productivos

6. Pruebas UI Automatizadas (Selenium + Pytest)

6.1 Descripción

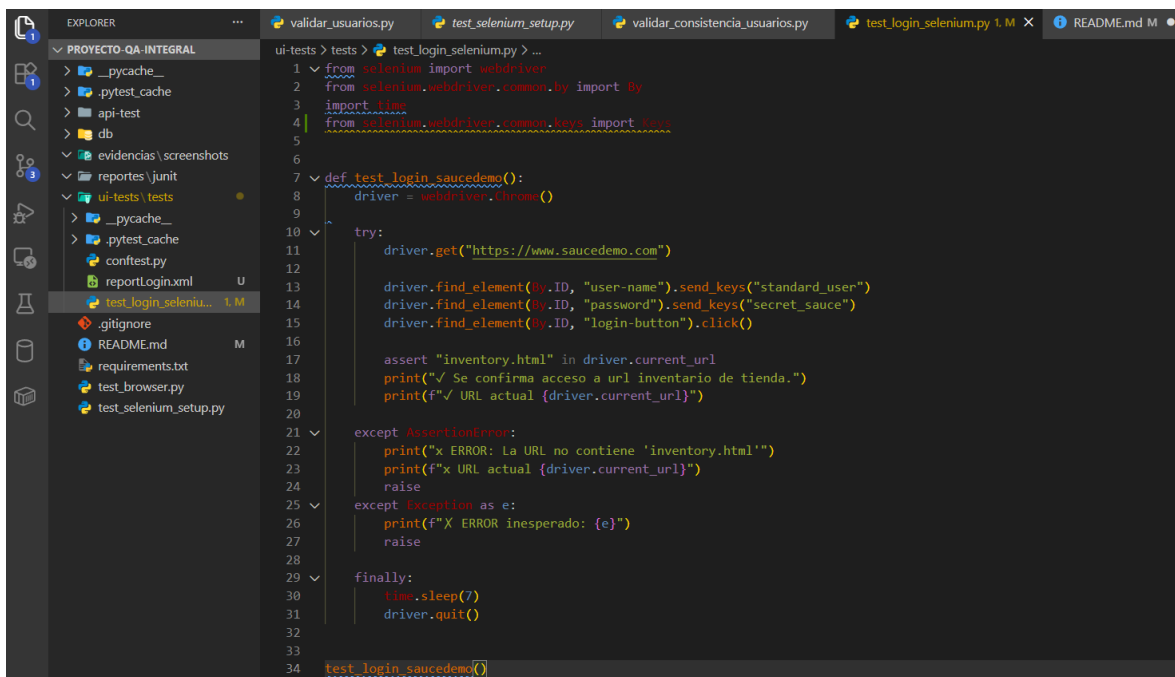
Se automatizó el flujo de login de una aplicación web utilizando Selenium WebDriver con Python y Pytest. El objetivo fue validar el acceso exitoso y comprobar la correcta redirección posterior al login.

El script incluye:

- Inicialización del navegador
- Interacción con elementos UI - DOM
- Aserciones funcionales
- Manejo básico de errores
- Captura de evidencias

6.2 Evidencias

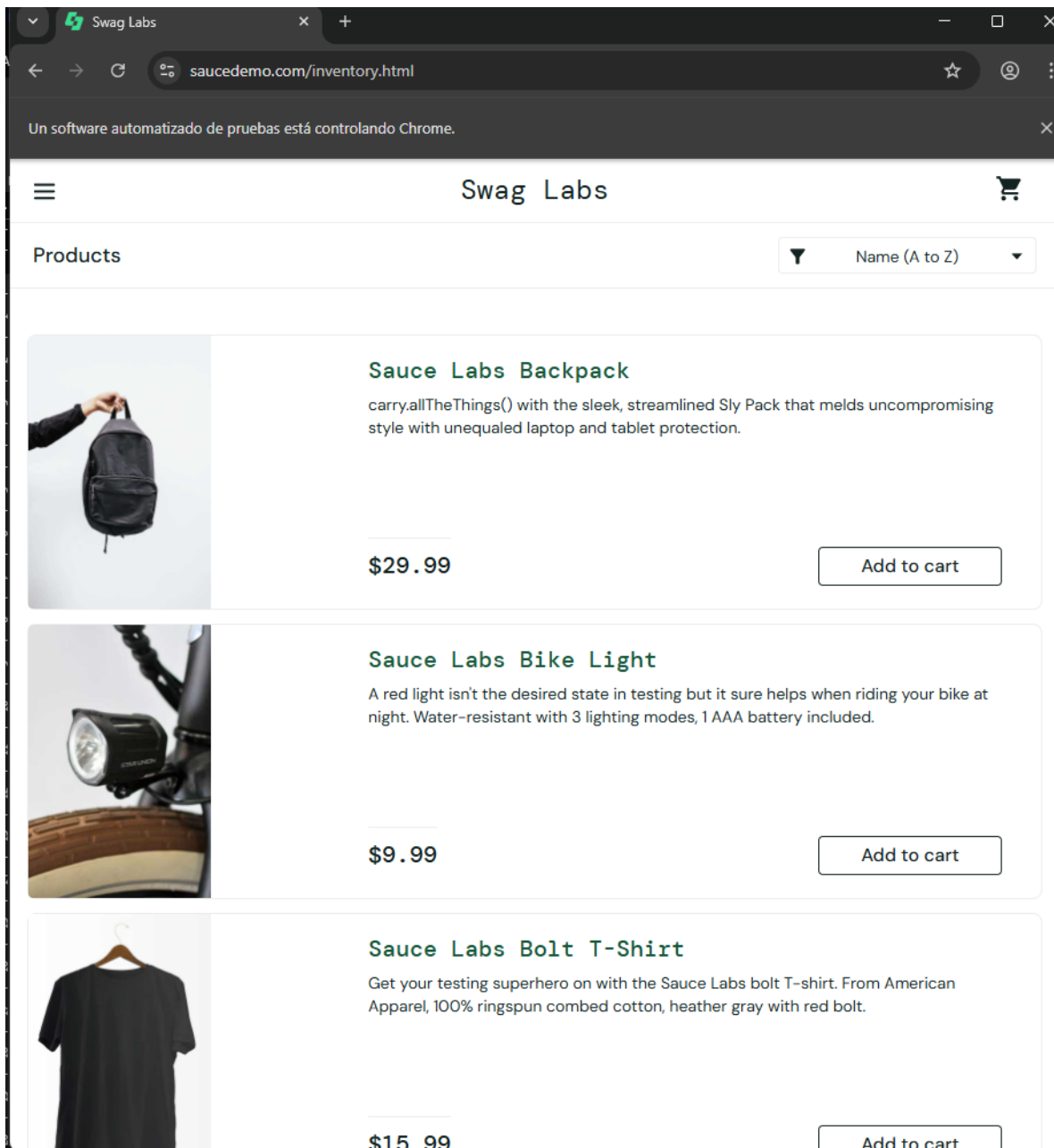
Script de automatización para login



```
EXPLORER
PROYECTO-QA-INTEGRAL
  __pycache__
  .pytest_cache
  api-test
  db
  evidencias\
    screenshots
  reportes\junit
  ui-tests\tests
    __pycache__
    .pytest_cache
    confest.py
    reportLogin.xml
    test_login_selenium.py
  .gitignore
  README.md
  requirements.txt
  test_browser.py
  test_selenium_setup.py

ui-tests > tests > test_login_selenium.py > ...
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 import time
4 from selenium.webdriver.common.keys import Keys
5
6
7 def test_login_saucedemo():
8     driver = webdriver.Chrome()
9
10    try:
11        driver.get("https://www.saucedemo.com")
12
13        driver.find_element(By.ID, "user-name").send_keys("standard_user")
14        driver.find_element(By.ID, "password").send_keys("secret_sauce")
15        driver.find_element(By.ID, "login-button").click()
16
17        assert "inventory.html" in driver.current_url
18        print("✓ Se confirma acceso a url inventario de tienda.")
19        print(f"✓ URL actual {driver.current_url}")
20
21    except AssertionError:
22        print("x ERROR: La URL no contiene 'inventory.html'")
23        print(f"x URL actual {driver.current_url}")
24        raise
25    except Exception as e:
26        print(f"x ERROR inesperado: {e}")
27        raise
28
29    finally:
30        time.sleep(7)
31        driver.quit()
32
33
34 test_login_saucedemo()
```

Captura de acceso a página con login ejecutado con script





Resultado usando Pytest en línea de comando

```
(venv) PS C:\QA Tester\proyecto-qa-integral> pytest .\ui-tests\tests\test_login_selenium.py -v
===== test session starts =====
platform win32 -- Python 3.13.3, pytest-9.0.2, pluggy-1.6.0 -- C:\venv_qa\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\QA Tester\proyecto-qa-integral
collected 1 item

ui-tests\tests\test_login_selenium.py::test_login_saucedemo PASSED [100%]

===== 1 passed in 22.92s =====
(venv) PS C:\QA Tester\proyecto-qa-integral>
```


Documento Junit

```
ui-tests > tests >  reporteLogin.xml >  xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <testsuites name="pytest tests">
3      <testsuite name="pytest" errors="0" failures="0" skipped="0" tests="1" time="23.440"
4          timestamp="2025-12-22T11:54:29.118419-03:00" hostname="SCL076PCAD005">
5          <testcase classname="test_login_selenium" name="test_login_saucedemo" time="11.555" />
6      </testsuite>
7  </testsuites>
```

6.3 Conocimientos aplicados

- Automatización UI con Selenium
- Uso de selectores - DOM
- Aserciones con Pytest
- Estructuración de pruebas automatizadas

6.4 Recomendaciones

- Implementar Page Object Model para mayor mantenibilidad
- Externalizar credenciales y datos sensibles
- Integrar ejecución en pipelines CI/CD

7. Gestión de Pruebas con Jira y Xray

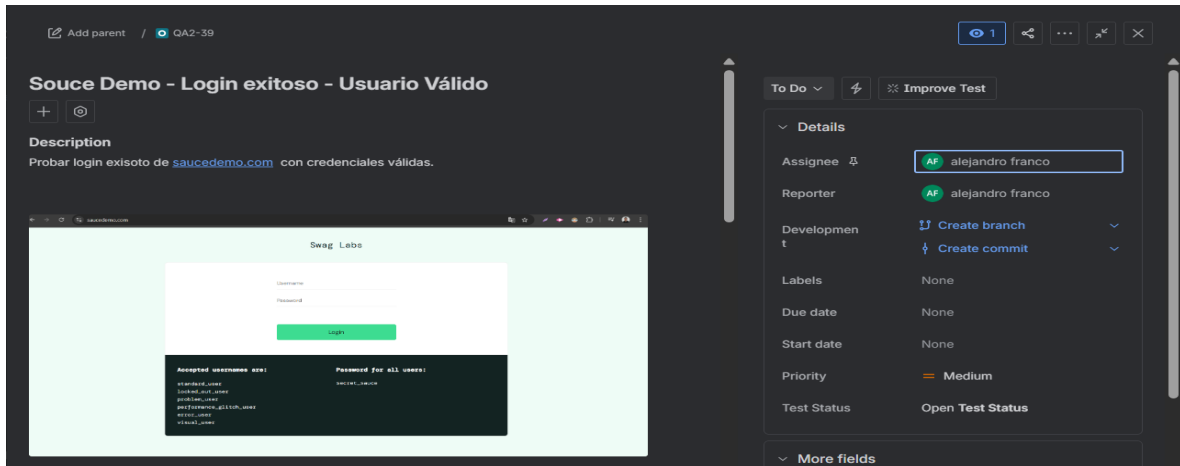
7.1 Descripción

Se utilizó Jira con el complemento Xray para:

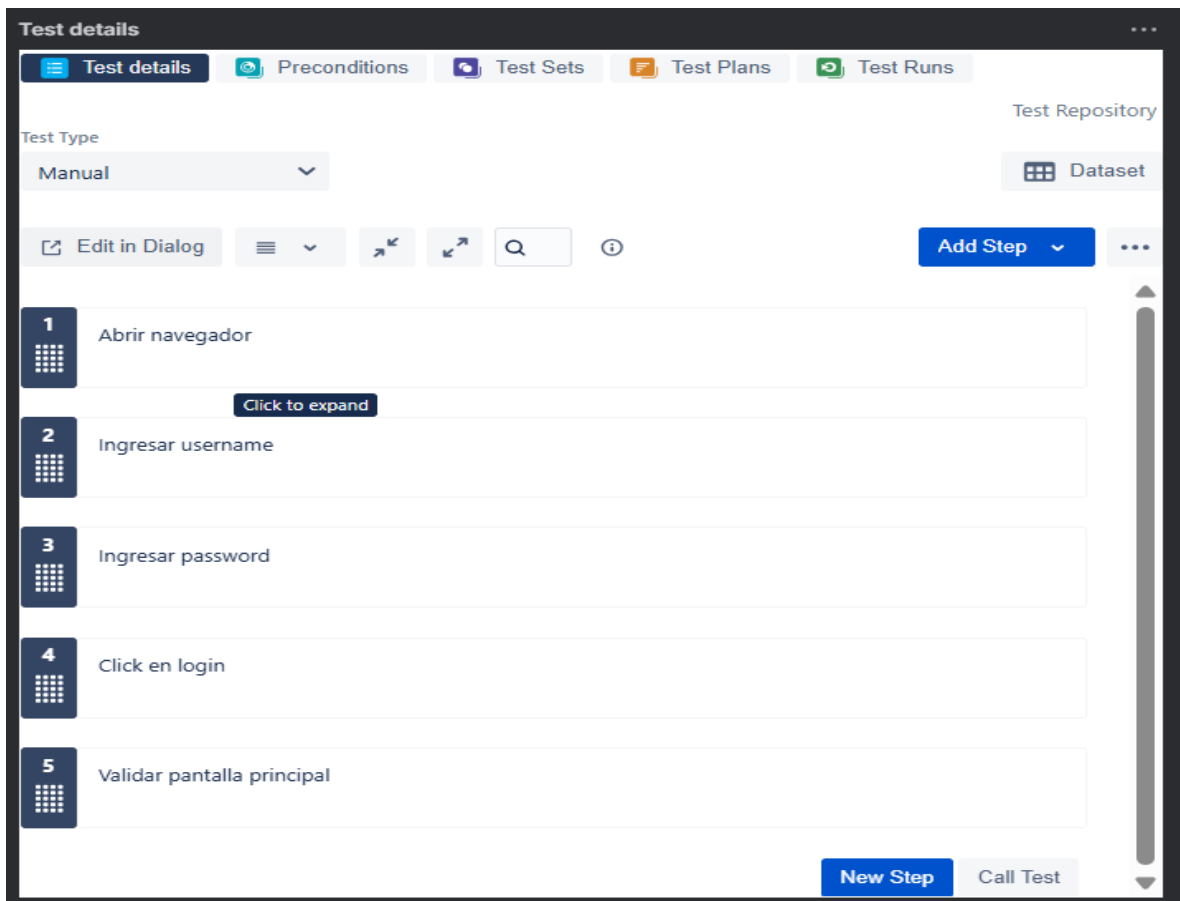
- Registrar casos de prueba manuales
- Registrar pruebas automatizadas
- Crear Test Plans y Test Executions
- Importar resultados automatizados vía XML (formato JUnit)
- Mantener trazabilidad entre tareas, tests y ejecuciones

7.2 Evidencias

Creación de Test con pasos detallados



Estructurar pasos de forma detallada y ordenada



Se crea *Test Execution* y agrega *Test Manual* con sus pasos para su ejecución

QA2 board

Projects / Quality Assurance 2 / Test Executions / QA2-46 / Test: QA2-39

Source Demo - Login exitoso - Usuario Válido

Test 1 of 1

Execute with Exploratory App | Dataset | Import Execution Results

Execution Status: **PASSED**

Timer: 00:00:12

Started On: 26/Dec/2025 12:38 PM

Finished On: 26/Dec/2025 12:38 PM

Assignee: Unassigned

Executed By: alejandro franco

Versions: -

Revision: -

Test Version: v1

Test Environments: -

No time logged

Findings: EVIDENCE (1)

Test details MANUAL

Test Description

Probar login exitoso de saucedemo.com con credenciales validas.

En la ejecución podemos registrar las evidencias según realicemos el paso a paso.

QA2-41 - navegador disponible

Steps

1	Action Abrir navegador	Data https://www.saucedemo.com/	Expected Result Página login visible	Step State PASSED
2	Action Ingresar username	Data Usuarios Registrados: standard_user	Expected Result Username ingresado	Step State PASSED
3	Action Ingresar password	Data Contraseña: secret_sauce	Expected Result Password ingresado	Step State PASSED
4	Action Click en login	Data Dato de referencia: id="login-button"	Expected Result Usuario autenticado	Step State PASSED
5	Action Validar pantalla principal	Data -	Expected Result <ul style="list-style-type: none">Dashboard visibleURL contiene saucedemo.com/inventory.htmlSe visualiza listado de productos	Step State PASSED

Activity

Se crea Test Execution y agrega XML de Test automatizado con Python

The screenshot shows the Jira Test Execution interface for a test execution titled "EXE - Automatizado - Login Exitoso". The interface is divided into several sections:

- Description:** A section for adding a description.
- Subtasks:** A section for adding subtasks.
- Linked work items:** A section for adding linked work items.
- Tests:** A section showing the overall execution status and a table of test results.
- Details:** A sidebar on the right showing details about the test execution, including Assignee, Reporter, Development, Labels, Due date, Start date, Priority, Test Plans, and Test Environments.
- More fields:** A section for adding more fields to the test execution.

The "Tests" section shows an "Overall Execution Status" bar that is green, indicating a "PASSED" status. Below this, there is a table with columns: Rank, Key, Summary, Test Type, Dataset, #Defects, and Status. The table contains one row with the following data:

Rank	Key	Summary	Test Type	Dataset	#Defects	Status
1	QA2-45	test_login_saucedemo	Generic		0	PASSED

The "Details" sidebar on the right shows the following information:

- Assignee:** alejandro franco
- Reporter:** alejandro franco
- Development:** Create branch, Create commit
- Labels:** None
- Due date:** None
- Start date:** None
- Priority:** Medium
- Test Plans:** Open Test Plans
- Test Environments:** Open Test Environments
- More fields:** Original estimate (0m), Time tracking (No time logged), Components (Add components), Sprint (None), Team (None)

7.3 Conocimientos aplicados

- Gestión de pruebas en Jira
- Uso de Xray para trazabilidad
- Importación de resultados automatizados (Documento estándar JUnit)
- Lectura e interpretación de reportes

7.4 Recomendaciones

- Definir convenciones de nombres para tests
- Centralizar evidencias en Test Executions
- Usar Test Plans para ciclos de prueba formales

8. Conclusiones

El Proyecto QA Integral permitió aplicar un enfoque completo de aseguramiento de calidad, abarcando múltiples capas del sistema y combinando pruebas manuales y automatizadas. Se adquirieron conocimientos prácticos en herramientas ampliamente utilizadas en la industria, así como en la gestión y trazabilidad de pruebas.

El proyecto deja una base sólida para evolucionar hacia prácticas más avanzadas como CI/CD, pruebas de performance y automatización a mayor escala.

Filosofía de Calidad

"El código no es evidencia. La evidencia es el resultado observable de la prueba: screenshots, logs y reportes. El script vive en Git, pero la evidencia de ejecución vive en Jira/Xray".