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)

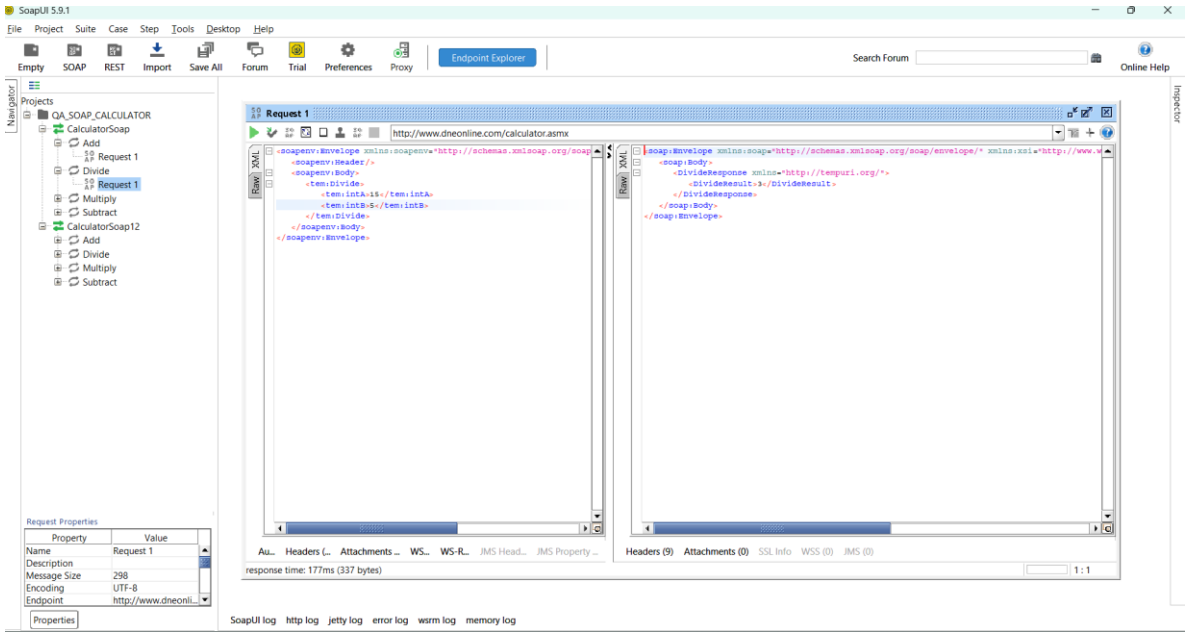
4.1 Descripción

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:

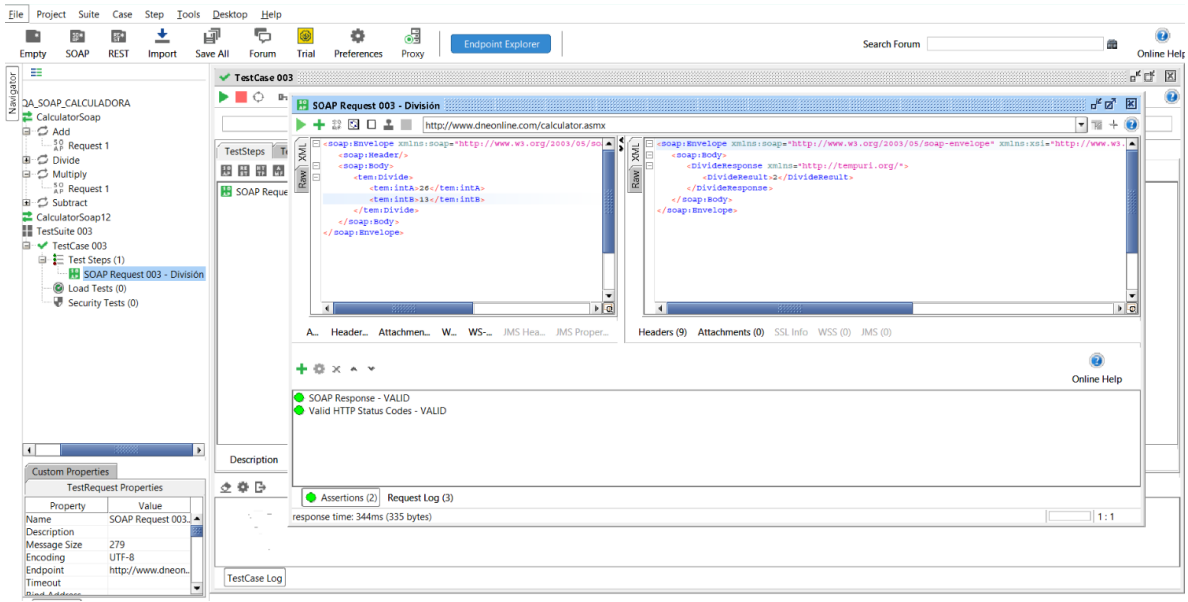
- Disponibilidad del servicio
- Código de respuesta HTTP
- Estructura del mensaje SOAP
- Resultado esperado de la operación

4.2 Evidencias

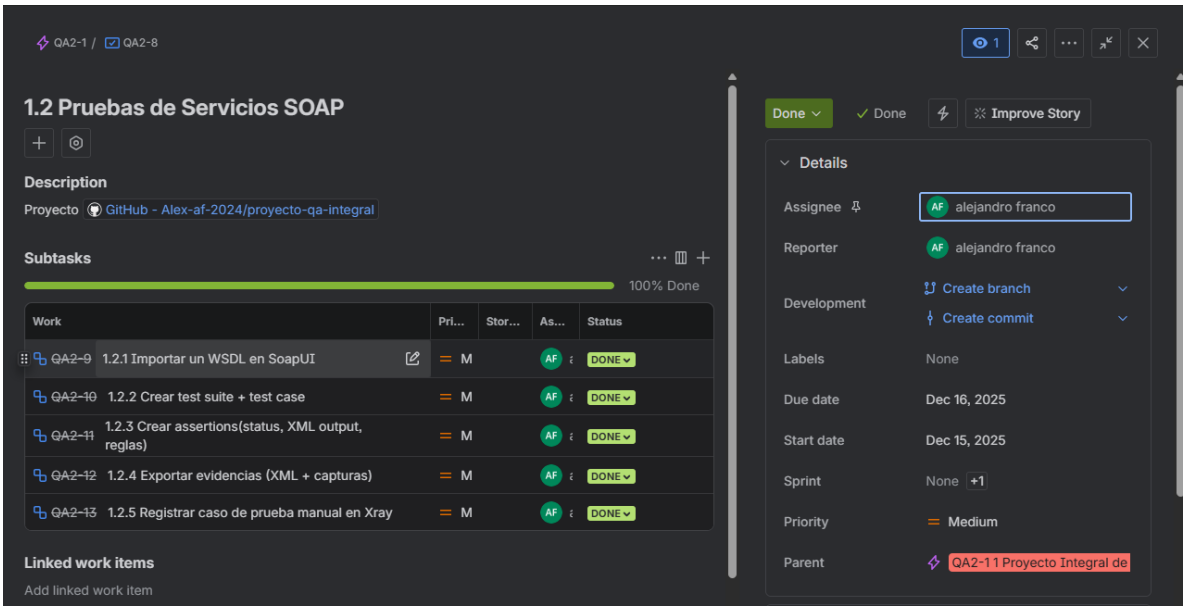
Request y Response de API en SoapUI (open source)



Test Suite, Test Case y Assertions(contains)



Se registra evidencias en Jira



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

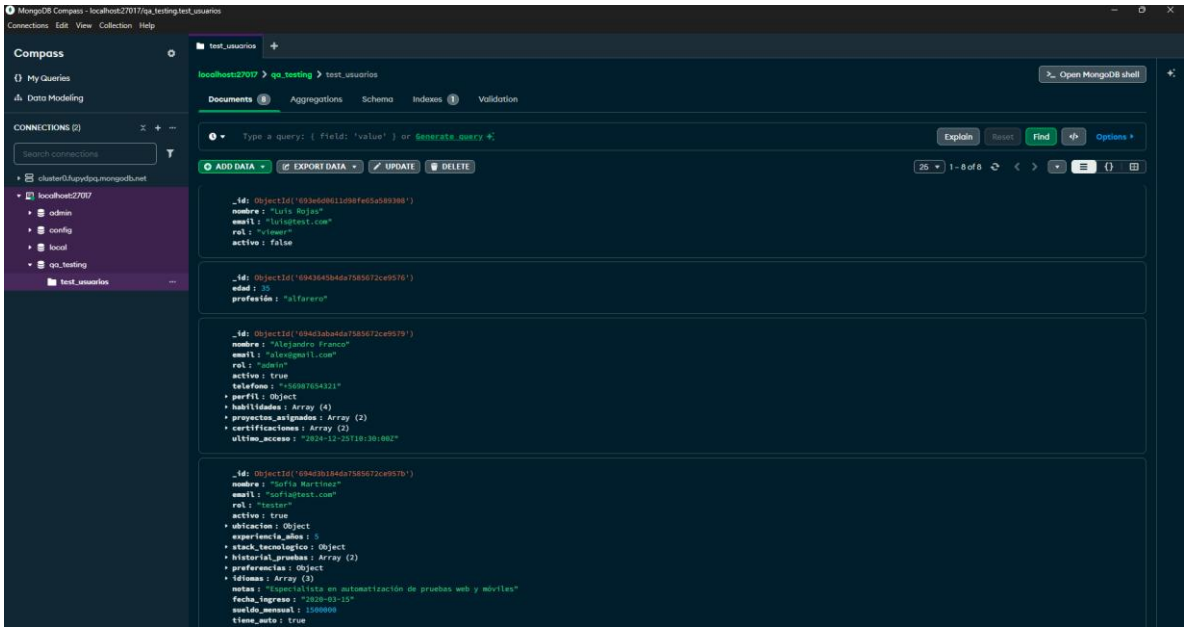
Se crea una base de datos de prueba en MongoDB para validar reglas básicas de integridad y consistencia de datos. Las validaciones se realizaron tanto de forma manual (MongoDB Compass) como automatizada mediante scripts en Python usando PyMongo.

Las validaciones incluyeron:

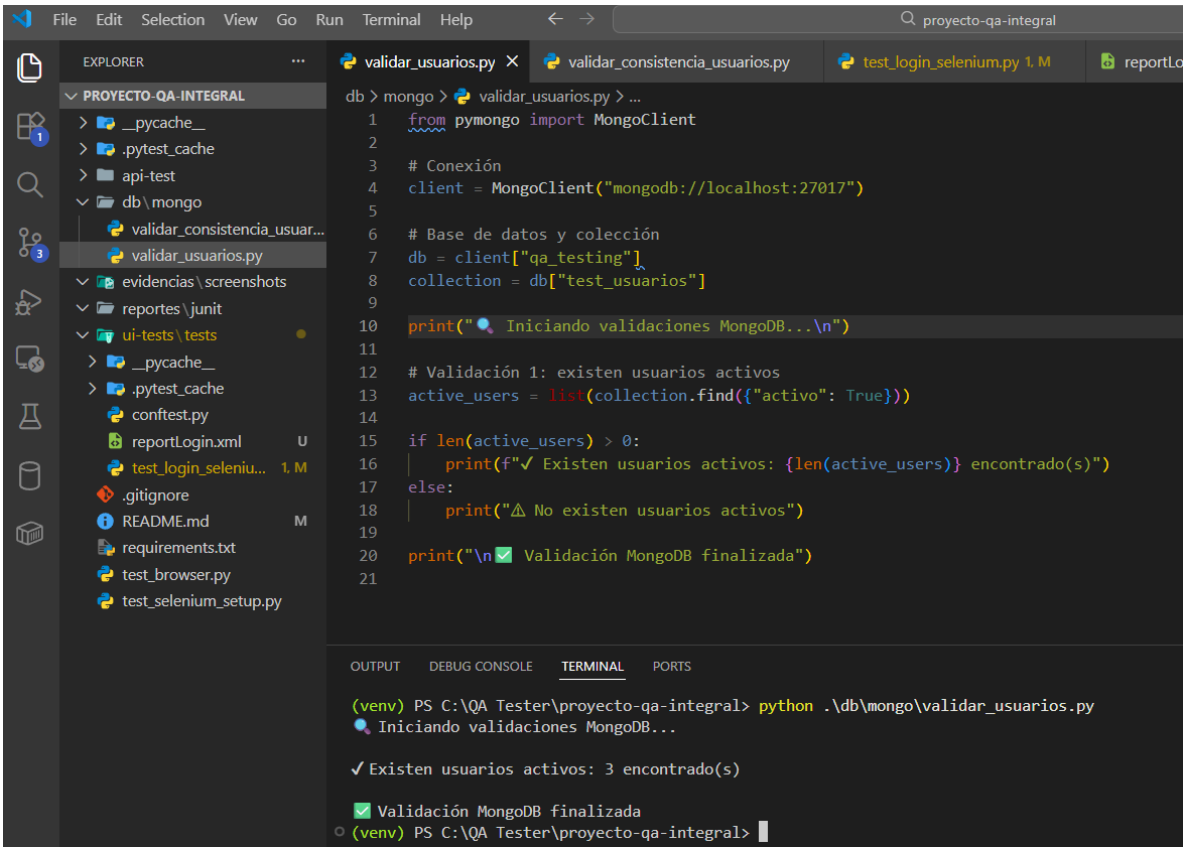
- Existencia de registros
- Validación de usuarios activos
- Validación de formato de emails
- Verificación de campos obligatorios

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



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

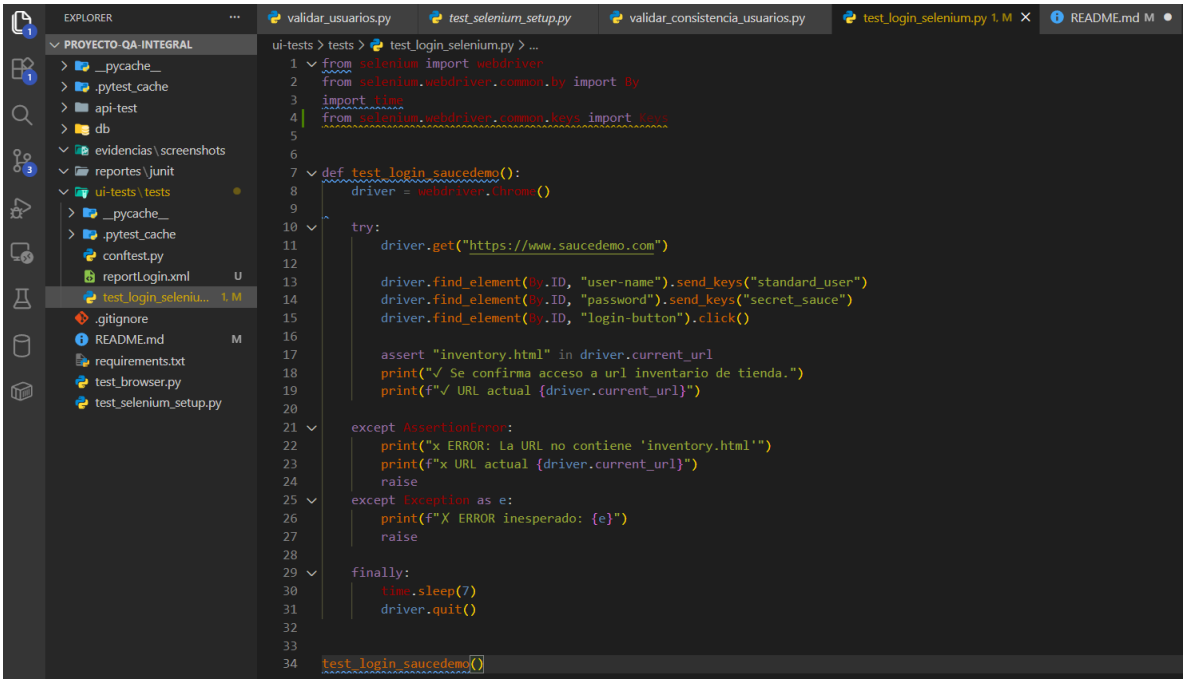
Se automatizó el flujo de login de una aplicación web utilizando Selenium WebDriver con Python y Pytest. El objetivo fue validar el acceso exitoso y comprobar la correcta redirección posterior al login.

El script incluye:

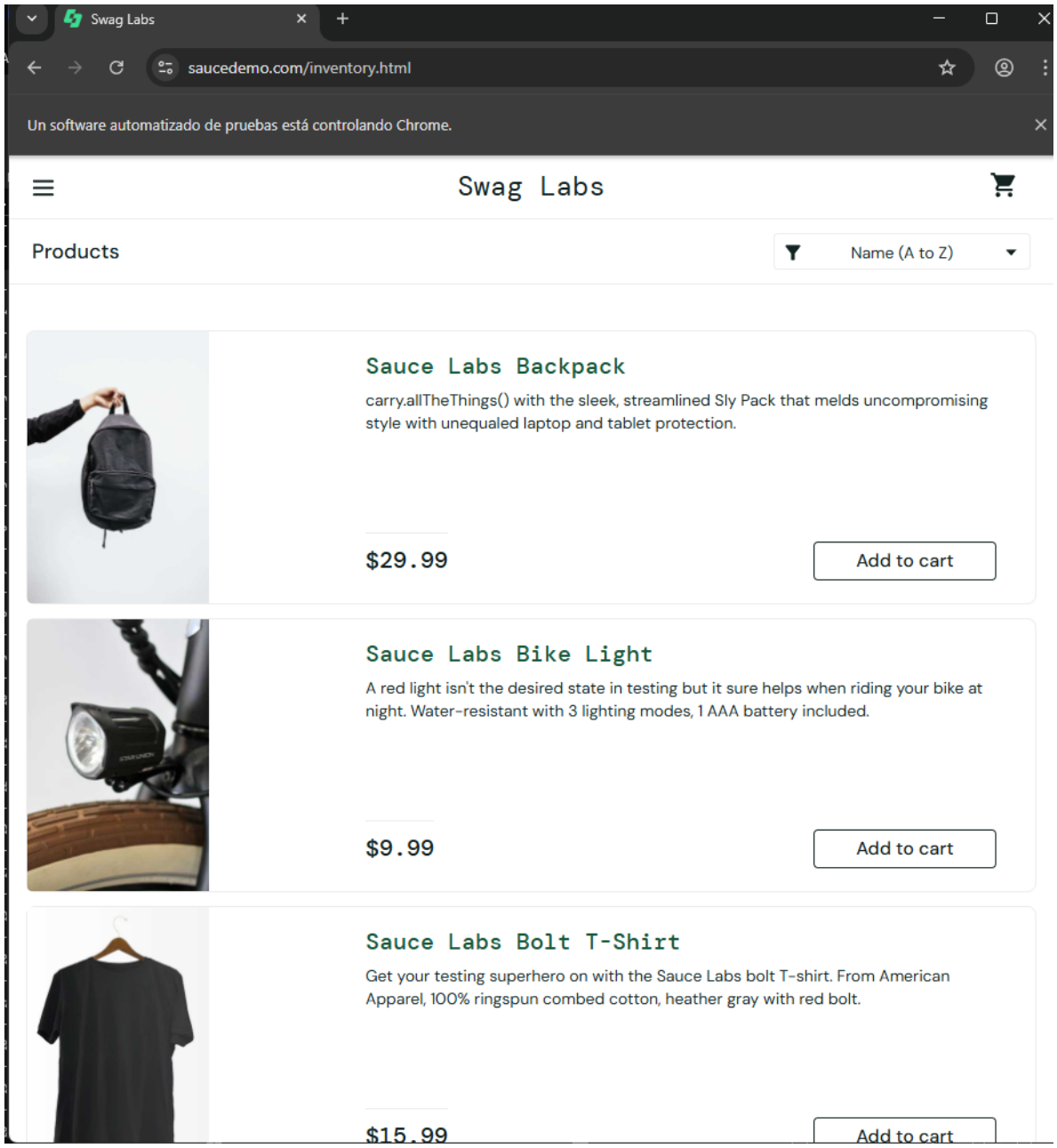
- Inicialización del navegador
- Interacción con elementos UI - DOM
- Aserciones funcionales
- Manejo básico de errores
- Captura de evidencias

6.2 Evidencias

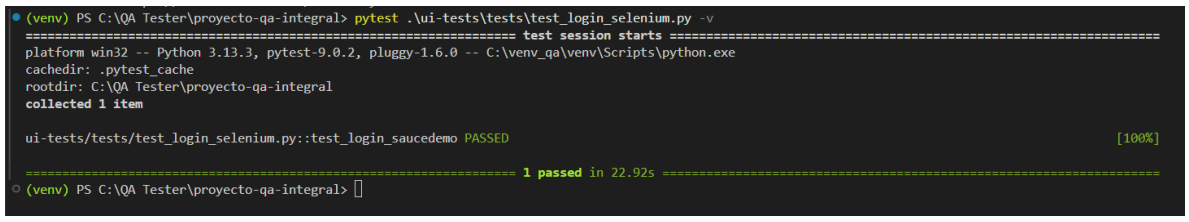
Script de automatización para login



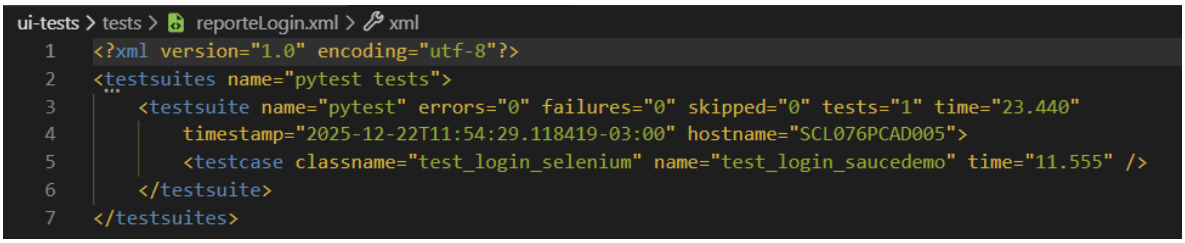
Captura de acceso a página con login ejecutado con script



Resultado usando Pytest en línea de comando



Documento Junit



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

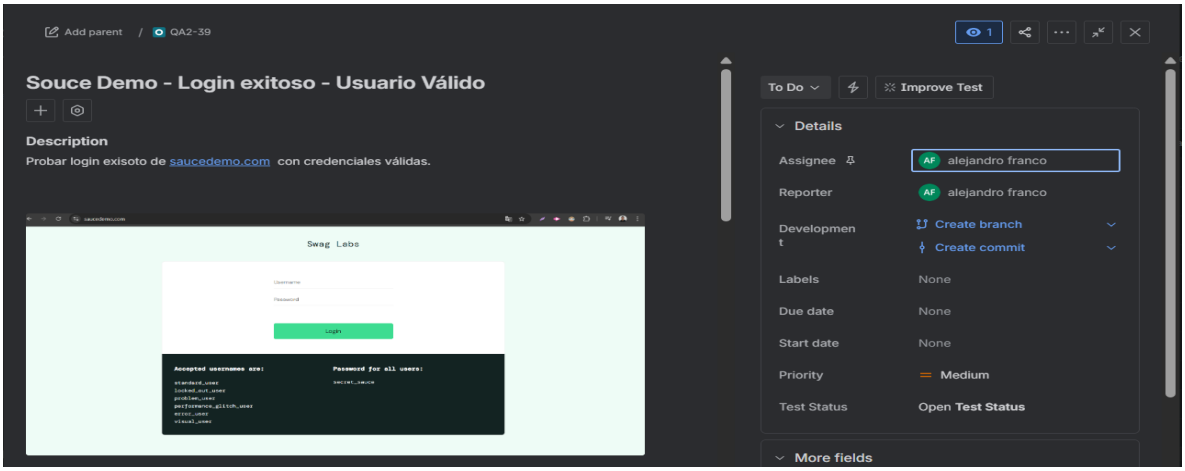
7.1 Descripción

Se utilizó Jira con el complemento Xray para:

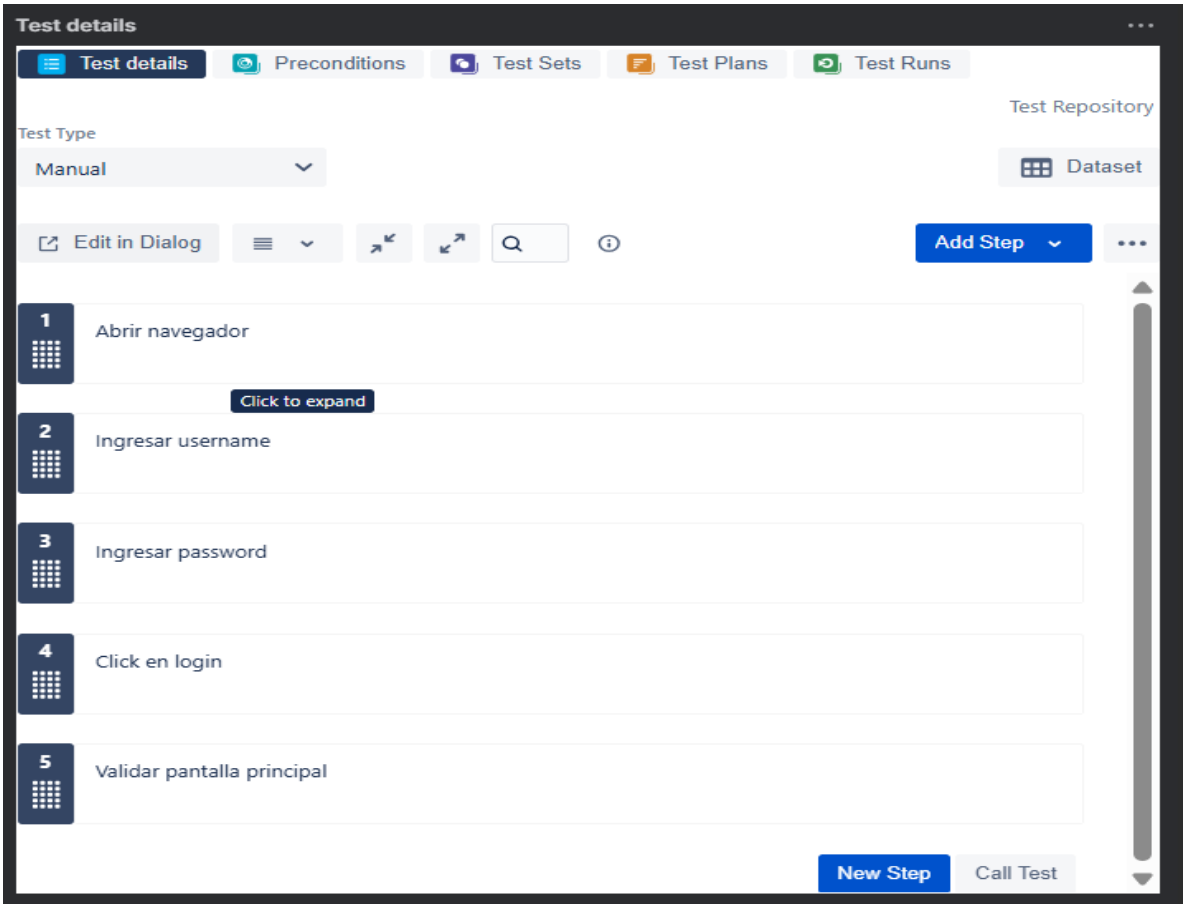
- Registrar casos de prueba manuales
- Registrar pruebas automatizadas
- Crear Test Plans y Test Executions
- Importar resultados automatizados vía XML (formato JUnit)
- Mantener trazabilidad entre tareas, tests y ejecuciones

7.2 Evidencias

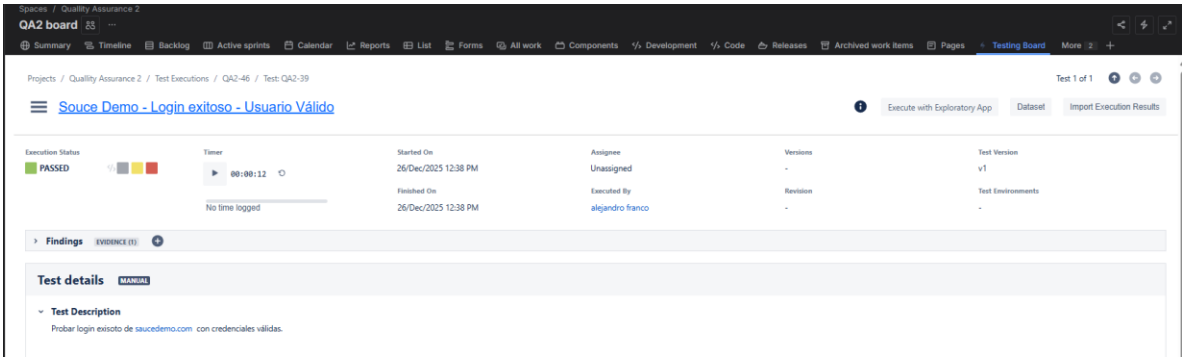
Creación de Test con pasos detallados



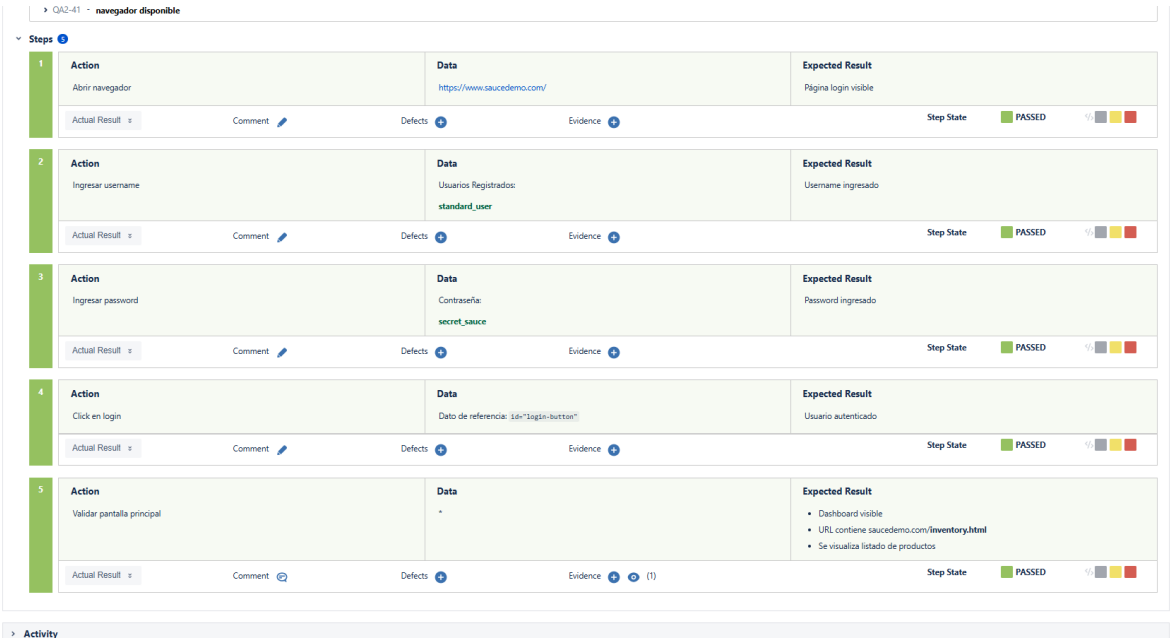
Estructurar pasos de forma detallada y ordenada



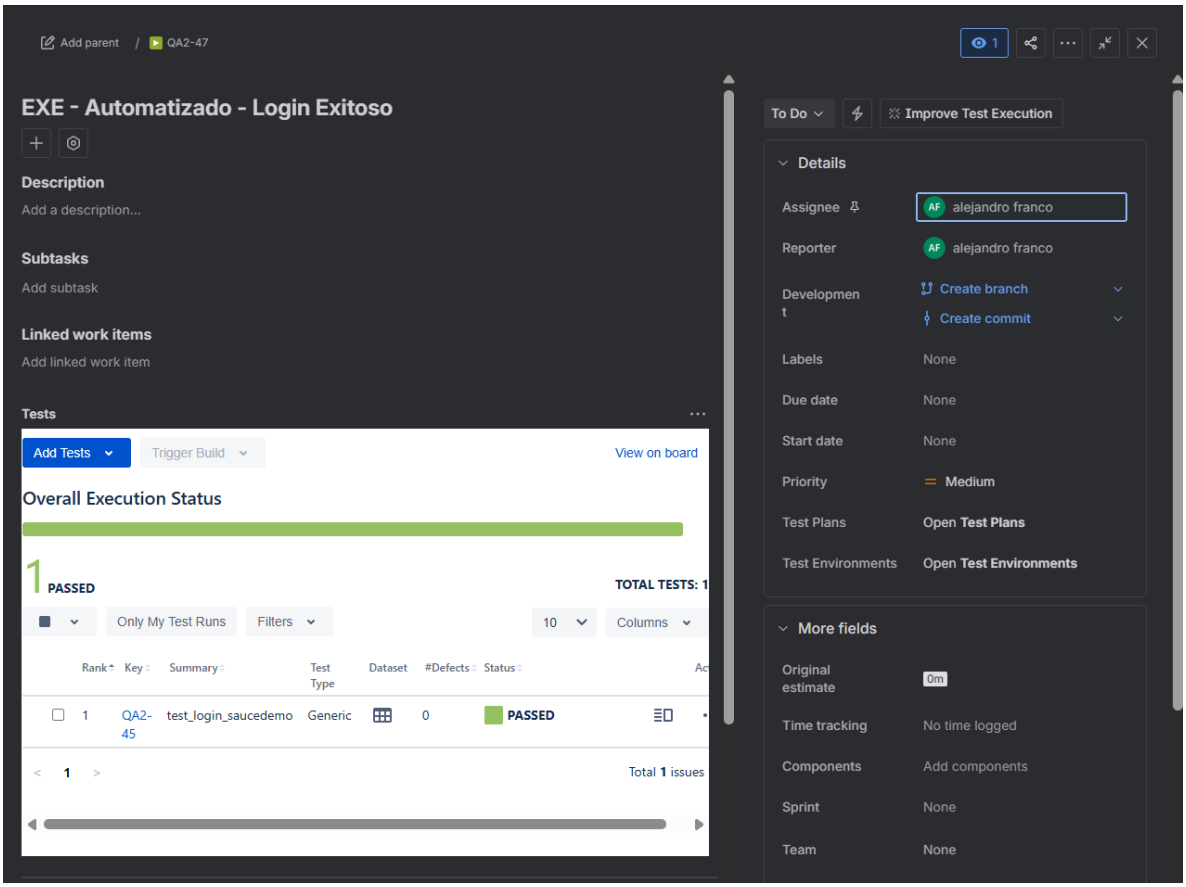
Se crea Test Execution y agrega Test Manual con sus pasos para su ejecución



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



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



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".