

Testing QA - Automation

Entrega de Proyecto Final

Alumno: Lucas Alvarez
Plataforma: SauceDemo (web)
URL: <https://www.saucedemo.com>
API Testeada: ReqRes (<https://reqres.in>)
Repositorio: [https://github.com/\[tu-usuario\]/proyecto-final-automation-testing-lucas-alvarez](https://github.com/[tu-usuario]/proyecto-final-automation-testing-lucas-alvarez)
Fecha de entrega: 22-11-2025

Índice

- 1. Descripción del Proyecto
 - 2. Tecnologías Utilizadas
 - 3. Estructura del Proyecto
 - 4. Pruebas de UI (Selenium)
 - 5. Pruebas de API (Requests)
 - 6. Page Object Model
 - 7. Parametrización de Datos
 - 8. Sistema de Reportes y Logging
 - 9. Resultados de Ejecución
 - 10. Instrucciones de Ejecución
 - 11. Conclusiones
-

1. Descripción del Proyecto

Este framework de automatización de pruebas fue desarrollado como proyecto final del curso de Testing QA - Automatización. El objetivo principal es validar la funcionalidad de una aplicación web (SauceDemo) y una API REST (ReqRes) utilizando las mejores prácticas de automatización.

Objetivos cumplidos:

- ☒ Automatización de flujos críticos de usuario (login, navegación, carrito, checkout)
 - ☒ Validación de endpoints de APIs REST (GET, POST, DELETE)
 - ☒ Implementación del patrón Page Object Model
 - ☒ Generación de reportes HTML detallados
 - ☒ Sistema de logging para depuración
 - ☒ Captura automática de screenshots en fallos
 - ☒ Parametrización de pruebas con datos externos (CSV)
-

2. Tecnologías Utilizadas

Tecnología	Versión	Propósito
------------	---------	-----------

Tecnología	Versión	Propósito
Python	3.13.7	Lenguaje principal
Pytest	9.0.1	Framework de testing
Selenium WebDriver	4.x	Automatización de UI
Requests	2.31+	Pruebas de API
pytest-html	4.1.1	Generación de reportes HTML
webdriver-manager	4.0+	Gestión automática de drivers
Git/GitHub	-	Control de versiones

3. Estructura del Proyecto

```
proyecto-final-automation-testing-lucas-alvarez/
├── datos/                                # Datos de prueba externos
│   └── user.csv                          # Credenciales para login parametrizado
├── logs/                                # Logs de ejecución (generado automático)
├── pages/                               # Page Object Model
│   ├── base_page.py                    # Clase base con métodos comunes
│   ├── login_page.py                  # Página de login
│   ├── inventory_page.py              # Página de productos
│   ├── cart_page.py                   # Página del carrito
│   ├── checkout_step_one_page.py
│   ├── checkout_step_two_page.py
│   └── checkout_complete_page.py
├── reports/                             # Reportes HTML
│   └── reporte.html
├── screenshots/                         # Capturas en fallos
├── tests/
│   ├── api/
│   │   └── test_api_reqres.py          # 7 tests de API
│   └── ui/
│       ├── test_login.py               # Login exitoso
│       ├── test_login_negativo.py      # 5 escenarios negativos
│       ├── test_login_parametrizado.py # Login con CSV
│       ├── test_cart.py                # Carrito
│       └── test_01_checkout.py          # Flujo E2E
├── utils/
│   ├── data_reader.py                 # Lectura de CSV/JSON
│   ├── helpers.py                     # Funciones auxiliares
│   └── logger.py                       # Sistema de logging
```

conftest.py

pytest.ini

requirements.txt

README.md

Fixtures de pytest

Configuración de pytest

Dependencias

Documentación

4. Pruebas de UI (Selenium)

4.1 Casos de Prueba Implementados

ID	Test	Descripción	Tipo	Resultado
UI-001	test_login_exitoso	Login con credenciales válidas	Positivo	✔ Pass
UI-002	test_login_usuario_invalido	Login con usuario inexistente	Negativo	✔ Pass
UI-003	test_login_password_incorrecta	Login con contraseña incorrecta	Negativo	✔ Pass
UI-004	test_login_campos_vacios	Login sin credenciales	Negativo	✔ Pass
UI-005	test_login_solo_usuario	Login solo con usuario	Negativo	✔ Pass
UI-006	test_login_usuario_bloqueado	Login con usuario bloqueado	Negativo	✔ Pass
UI-007	test_login_parametrizado[fila0]	Login standard_user (CSV)	Parametrizado	✔ Pass
UI-008	test_login_parametrizado[fila1]	Login locked_out_user (CSV)	Parametrizado	✔ Pass
UI-009	test_login_parametrizado[fila2]	Login problem_user (CSV)	Parametrizado	✔ Pass
UI-010	test_login_parametrizado[fila3]	Login fake_user (CSV)	Parametrizado	✔ Pass
UI-011	test_agregar_producto_al_carrito	Agregar producto y verificar carrito	Positivo	✔ Pass
UI-012	test_checkout_completo	Flujo E2E de compra completa	E2E	⚠ Intermitente

4.2 Flujos Cubiertos

- 1. **Flujo de Login:** Validación completa con escenarios positivos y negativos
- 2. **Flujo de Carrito:** Agregar productos y verificar contenido

3. **Flujo de Checkout:** Login → Inventario → Carrito → Checkout → Confirmación

5. Pruebas de API (Requests)

5.1 API Testeada: ReqRes (https://reqres.in)

ID	Test	Método	Endpoint	Descripción	Resultado
API-001	test_get_lista_usuarios	GET	/users?page=2	Obtener lista paginada	✔ Pass
API-002	test_get_usuario_individual	GET	/users/2	Obtener usuario específico	✔ Pass
API-003	test_crear_usuario	POST	/users	Crear nuevo usuario	✔ Pass
API-004	test_crear_y_eliminar_usuario	POST + DELETE	/users	Encadenamiento de peticiones	✔ Pass
API-005	test_usuario_no_encontrado	GET	/users/9999	Validar 404	✔ Pass
API-006	test_login_exitoso	POST	/login	Login simulado	✔ Pass
API-007	test_login_fallido_sin_password	POST	/login	Login sin password (400)	✔ Pass

5.2 Validaciones Realizadas

- ✔ Códigos de estado HTTP (200, 201, 204, 400, 404)
- ✔ Estructura de respuestas JSON
- ✔ Contenido de campos específicos
- ✔ Encadenamiento de peticiones (crear → eliminar)

6. Page Object Model

6.1 Implementación

El proyecto implementa el patrón Page Object Model para separar la lógica de pruebas de la interacción con la UI.

Clase Base (base_page.py):

```
class BasePage:
    def __init__(self, driver):
        self.driver = driver

    def navegar(self, url):
```

```
self.driver.get(url)

def encontrar(self, locator, timeout=10):
    return WebDriverWait(self.driver, timeout).until(
        EC.visibility_of_element_located(locator)
    )

def click(self, locator, timeout=10):
    self.encontrar(locator, timeout).click()

def escribir(self, locator, texto, timeout=10):
    campo = self.encontrar(locator, timeout)
    campo.clear()
    campo.send_keys(texto)
```

6.2 Páginas Implementadas

Página	Archivo	Métodos Principales
Login	login_page.py	abrir(), login(), obtener_mensaje_error()
Inventario	inventory_page.py	agregar_producto_backpack(), ir_al_carrito()
Carrito	cart_page.py	obtener_titulo(), ir_a_checkout()
Checkout Paso 1	checkout_step_one_page.py	completar_formulario(), continuar()
Checkout Paso 2	checkout_step_two_page.py	finalizar()
Checkout Completo	checkout_complete_page.py	obtener_mensaje_final()

7. Parametrización de Datos

7.1 Archivo de Datos (datos/user.csv)

```
username,password,resultado
standard_user,secret_sauce,ok
locked_out_user,secret_sauce,error
problem_user,secret_sauce,ok
fake_user,123,error
```

7.2 Implementación en Test

```
from utils.data_reader import leer_csv

datos = leer_csv("user.csv")

@pytest.mark.parametrize("fila", datos)
def test_login_parametrizado(browser, fila):
```

```
login = LoginPage(browser)
login.abrir()
login.login(fila["username"], fila["password"])

if fila["resultado"] == "ok":
    assert "inventory" in browser.current_url
else:
    assert "Epic sadface" in login.obtener_mensaje_error()
```

8. Sistema de Reportes y Logging

8.1 Reportes HTML (pytest-html)

- **Ubicación:** `reports/reporte.html`
- **Contenido:** Resumen de ejecución, detalle de cada test, logs capturados
- **Generación automática:** Configurado en `pytest.ini`

8.2 Sistema de Logging

Cada ejecución genera logs individuales por test en la carpeta `logs/`:

```
logs/
├─ test_login_exitoso_2025-11-21_18-09-31.log
├─ test_login_parametrizado[fila0]_2025-11-21_18-10-35.log
└─ ...
```

8.3 Captura Automática de Screenshots

Configurado en `conftest.py` para capturar automáticamente cuando una prueba falla:

```
if hasattr(request.node, "rep_call") and request.node.rep_call.failed:
    logger.error("[FAILED] La prueba FALLO - Generando captura...")
    _capture_screenshot(driver, request, logger)
```

9. Resultados de Ejecución

9.1 Resumen General

Métrica	Valor
Total de tests	19
Tests pasados	18
Tests fallidos	1

Métrica	Valor
Tasa de éxito	94.7%
Tiempo total	02:34

9.2 Resultados por Categoría

Categoría	Total	Pasados	Fallidos
Pruebas de API	7	7	0
Pruebas de UI	12	11	1*

*El test de checkout completo presenta inestabilidad intermitente debido a problemas de sincronización con Chrome.

9.3 Detalle de Ejecución

```
tests/api/test_api_reqres.py ..... [ 36%]
tests/ui/test_01_checkout.py F [ 42%]
tests/ui/test_cart.py . [ 47%]
tests/ui/test_login.py . [ 52%]
tests/ui/test_login_negativo.py ..... [ 78%]
tests/ui/test_login_parametrizado.py .... [100%]

===== 1 failed, 18 passed in 209.85s =====
```

10. Instrucciones de Ejecución

10.1 Prerrequisitos

- Python 3.8 o superior
- Google Chrome instalado
- Git

10.2 Instalación

```
# Clonar repositorio
git clone https://github.com/[tu-usuario]/proyecto-final-automation-testing-lucas-alvarez.git
cd proyecto-final-automation-testing-lucas-alvarez

# Crear entorno virtual
python -m venv venv

# Activar entorno (Windows)
venv\Scripts\activate
```

```
# Instalar dependencias
pip install -r requirements.txt
```

10.3 Ejecución de Pruebas

```
# Ejecutar todas las pruebas con reporte
pytest --html=reports/reporte.html --self-contained-html

# Solo pruebas de UI
pytest tests/ui/ -v

# Solo pruebas de API
pytest tests/api/ -v

# Test específico
pytest tests/ui/test_login.py -v
```

11. Conclusiones

11.1 Logros del Proyecto

1. **Framework completo y funcional** con arquitectura mantenible
2. **Cobertura de pruebas UI y API** cumpliendo los requerimientos
3. **Implementación correcta de POM** facilitando mantenimiento
4. **Sistema de reportes profesional** con logs detallados
5. **Parametrización efectiva** usando fuentes externas de datos

11.2 Áreas de Mejora Futura

- Integración con CI/CD (GitHub Actions)
- Ejecución en paralelo para reducir tiempos
- Mejora de estabilidad en pruebas E2E
- Integración con Allure Reports para reportes más visuales

11.3 Aprendizajes

Este proyecto permitió aplicar de manera práctica los conceptos de automatización de pruebas, incluyendo patrones de diseño, manejo de datos externos, generación de reportes y buenas prácticas de código mantenible.

Repositorio GitHub: [Insertar URL del repositorio]

Autor: Lucas Alvarez

Curso: Testing QA - Automatización

Fecha: Noviembre 2025