

Test Plan – Swag Labs

1. Visión General

1.1 Introducción

El presente documento define el **Plan de Pruebas (Test Plan)** para el proyecto Swag Labs, alojado en <https://www.saucedemo.com/>, con propósito de realizar el challenge solicitado

Este plan es una guía para la ejecución de actividades de calidad orientadas al challenge propuesto.

1.2 Propósito

El propósito de este Test Plan es:

- Establecer estrategia, alcance, recursos, herramientas, criterios de entrada/salida y cronograma de testing.
- Asegurar que los entregables cumplen con los requisitos definidos.

1.2 Objetivo

El objetivo de este plan es validar la funcionalidad completa de Inicio de Sesión y Carrito de Compras de la web Swag Labs, asegurando que el sistema funcione correctamente según los requisitos funcionales y criterios de aceptación, detectando defectos y mitigando riesgos antes de producción.

2. Conformación del Equipo

Rol	Nombre	Responsabilidades
QA Engineer	Matias Murua	Diseño y ejecución de casos de prueba, automatización de pruebas, documentación y registro de defectos.

3. Estrategia

3.1 Tipos de pruebas a ejecutar

- 1. **Smoke Tests** — Validación básica de build/deployment.
- 2. **Functional Testing** — Validación funcional completa.
- 3. **Integration Testing** — APIs, servicios externos, backend–frontend.
- 4. **Automation Testing** — diseñar framework de automatización para la ejecución de las pruebas solicitadas

3.2 Herramientas de testing

Categoría	Herramienta
Gestión de pruebas	Documento de google
Automatización UI	Cypress
Automatización API	Cypress
Versionado	Git

4. Casos de prueba

Iniciar sesión

ID	NOMBRE
TC-LOGIN-001	Login válido con standard_user
TC-LOGIN-002	Login válido con problem_user
TC-LOGIN-003	Login válido con performance_glitch_user
TC-LOGIN-004	Login con usuario bloqueado
TC-LOGIN-005	Login con password incorrecta
TC-LOGIN-006	Login con usuario incorrecto
TC-LOGIN-007	Login con ambos campos incorrectos
TC-LOGIN-008	Login con username vacío
TC-LOGIN-009	Login con password vacío
TC-LOGIN-010	Login con ambos campos vacíos

TC-LOGIN-011	Login con espacios en username
TC-LOGIN-012	Login con espacios en password
TC-LOGIN-013	Validar case sensitivity username
TC-LOGIN-014	Validar case sensitivity password
TC-LOGIN-015	Login presionando Enter
TC-LOGIN-016	Validar múltiples intentos fallidos
TC-LOGIN-017	Login con caracteres especiales
TC-LOGIN-018	Validar redirección post login
TC-LOGIN-019	Validar botón Login habilitado

Carrito de Compras

TC-CART-001	Agregar producto al carrito desde Products
TC-CART-002	Validar badge del carrito actualizado
TC-CART-003	Visualizar producto en el carrito
TC-CART-004	Validar información correcta del producto
TC-CART-005	Eliminar producto del carrito
TC-CART-006	Validar carrito vacío
TC-CART-007	Validar botón Continuar comprando
TC-CART-008	Validar botón Checkout
TC-CART-009	Completar checkout con datos validos
TC-CART-010	Finalizar compra
TC-CART-011	Validar cálculo precio total
TC-CART-012	Checkout sin First Name
TC-CART-013	Checkout sin Last Name
TC-CART-014	Checkout sin Zip Code
TC-CART-015	Login presionando Enter
TC-CART-016	Validar múltiples intentos fallidos
TC-CART-017	Login con caracteres especiales
TC-CART-018	Validar redirección post login
TC-CART-019	Validar botón Login habilitado
TC-CART-020	Checkout con todos campos vacíos
TC-CART-021	Acceder checkout sin productos
TC-CART-022	Validar eliminación actualiza badge
TC-CART-024	Validar botón Cancel en Overview
TC-CART-025	Intentar finalizar sin productos
TC-CART-026	Validar múltiples productos checkout

4.1 Detalle casos de prueba

ID	Nombre	Descripción	Precondición	Pasos a seguir	Resultado esperado
TC-LOGIN-001	Login válido con standard_user	Verificar que un usuario válido puede iniciar sesión	Usuario en página de login	1. Ingresar username: standard_user 2. Ingresar password: secret_sauce 3. Click en Login	El usuario inicia sesión correctamente y es redirigido a la página de inicio de productos
TC-LOGIN-004	Login con usuario bloqueado	Verificar comportamiento con usuario bloqueado	Usuario en página login	1. Ingresar locked_out_user 2. Ingresar secret_sauce 3. Click Login	Se muestra el siguiente mensaje de error: "Epic sadface: Sorry, this user has been locked out."
TC-LOGIN-005	Login con password incorrecta	Verificar que no se permita login ingresando una password incorrecta	Usuario en página login	1. Ingresar standard_user 2. Ingresar password incorrecto 3. Click Login	Se muestra el siguiente mensaje de error: "Epic sadface: Username and password do not match any user in this service"
TC-CART-027	Flujo completo de compra exitosa	Validar que un usuario pueda completar exitosamente todo el proceso de compra desde agregar un producto hasta visualizar la confirmación final	Usuario logueado en Swag Labs con credenciales válidas	1. Hacer click en "Add to cart" en un producto 2. Verificar que el badge del carrito muestra "1" 3. Hacer click en el ícono del carrito 4. Verificar que el producto esté listado correctamente 5. Hacer click en el botón "Checkout" 6. Completar el campo "First Name" con un valor válido 7. Completar el campo "Last Name" con un valor válido 8. Completar el campo "Zip Code" con un valor válido 9. Hacer click en el botón "Continue" 10. Verificar que se muestre el resumen de compra correctamente 11. Hacer click en el botón "Finish"	El sistema muestra la pantalla de confirmación con el mensaje ***"Thank you for your order!"***, indicando que la compra fue completada exitosamente
TC-CART-001	Agregar	Validar que un	Usuario	1. Ir a Products	El producto se agrega al

	producto al carrito desde Products	usuario pueda agregar un producto al carrito	logueado en Swag Labs	2. Hacer click en "Add to cart" en un producto	carrito y el ícono muestra badge "1"
TC-CART-009	Completar checkout con datos válidos	Validar checkout exitoso	Producto en carrito	1. Click Checkout 2. Completar First Name 3. Completar Last Name 4. Completar Zip Code 5. Click Continue	Permite dirigirse a la página Checkout Overview para poder observar el detalle de la compra.

5. Definición Test Suite

Smoke Test Suit

TC-LOGIN-001 — Login válido con standard_user
 TC-LOGIN-018 — Validar redirección post login
 TC-CART-001 — Agregar producto al carrito
 TC-CART-003 — Visualizar producto en carrito
 TC-CART-008 — Validar botón Checkout
 TC-CART-009 — Completar checkout correctamente
 TC-CART-010 — Finalizar compra
 TC-CART-027 — Flujo completo de compra exitosa

La Suite de Smoke contiene los casos críticos. El objetivo es verificar que las funciones principales del sistema (el "camino feliz") operen correctamente tras un nuevo despliegue. Si un Smoke falla, el sistema no es estable para seguir probando.

Regression Test Suit

Podríamos agregar los demás casos de prueba restantes ya que la Suite de Regresión es exhaustiva. Su objetivo es confirmar que los cambios recientes o correcciones de bugs no hayan afectado negativamente a funcionalidades existentes. Aquí incluimos casos de borde, validaciones de errores y comportamientos específicos.

6. Defectos

Se aplicará el siguiente flujo de defectos:

1. Nuevo
2. Asignado
3. En Desarrollo
4. En Verificación QA
5. Reabierto / Resuelto
6. Cerrado

Cada defecto deberá incluir:

- Severidad y prioridad
- Pasos para reproducir
- Evidencias (screenshots / logs)

6.1 Defecto encontrado

Nombre:

Usuario problem_user no puede remover productos del carrito luego de agregarlos

Descripción:

Se detecta que al iniciar sesión con el usuario problem_user, el sistema permite agregar productos al carrito correctamente, pero el botón "Remove" no elimina el producto agregado. Este comportamiento rompe la funcionalidad principal del carrito de compras, ya que el usuario no puede gestionar ni eliminar productos agregados, afectando directamente el flujo de compra.

Severidad: Alta (High)

Impacta una funcionalidad crítica del sistema (gestión del carrito), bloqueando parcialmente el flujo de compra.

Prioridad: Alta (High)

Debe resolverse con prioridad alta porque afecta una funcionalidad core del negocio y la experiencia del usuario

Precondiciones:

Usuario registrado en el sistema

Usuario válido: problem_user

Password válida : secret_sauce

Usuario autenticado correctamente

Aplicación disponible: <https://www.saucedemo.com/>

Pasos para reproducir:

1-Navegar a <https://www.saucedemo.com/>

2-Ingresa el username: problem_user

3-Ingresa el password: secret_sauce

4-Hacer click en el botón Login

5-En la página Products, hacer click en Add to cart en el producto "Sauce Labs Backpack"

6-Verificar que el botón cambia a Remove

7-Hacer click en el botón Remove

Resultado esperado:

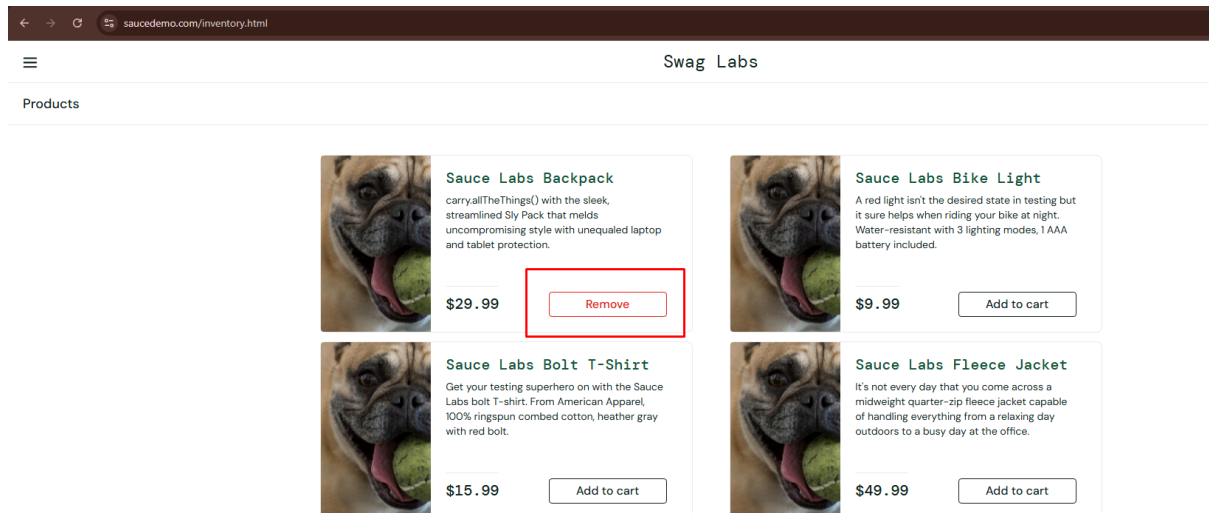
El producto debe eliminarse del carrito y el botón debe cambiar nuevamente a Add to cart.

Resultado actual:

El producto no se elimina del carrito y el botón Remove no tiene efecto.

Evidencias:

Screenshot adjunto:



Se observa que el botón Remove está visible pero no ejecuta ninguna acción.

7. Pasos para realizar pruebas API:

1-Analizar la documentación y requerimientos:

Analizar criterios de aceptación solicitados, luego revisar documentación en Swagger para entender el funcionamiento de la api.

2-Identificar escenarios de prueba

Definir los distintos escenarios que deben validarse (casos positivos, negativos, edge, seguridad)

3-Diseñar los casos de prueba:

Los casos de prueba a diseñar deben incluir:

ID del caso

Endpoint y método HTTP

Headers

Body (si aplica)

Resultado esperado

Código de estado esperado

Validaciones del response

4-Preparar el entorno:

Definir base_url del entorno

Preparar datos de prueba

Verificar que el servicio esté disponible

5-Ejecutar las pruebas:

Realizar solicitudes a la Api usando herramientas como Postman validando:

Código de estado (status code)

Estructura del response

Datos retornados

Tiempo de respuesta
Cumplimiento de las reglas de negocio

6-Analizar resultados y reportar defectos:
Comparar el resultado obtenido con el esperado:
Si es correcto se marca PASSED
Si es incorrecto, se reporta un bug con evidencia

8. Ejemplo Caso de prueba API

ID: TC-API-LOGIN-001

Nombre: Validar login exitoso con credenciales válidas

Descripción: Verificar que el sistema permite autenticarse con credenciales correctas

Request:

Método: POST

Endpoint: /api/login

Headers:

Accept : application/json

Content-Type: application/json

Body:

```
{  
  "username": "standard_user",  
  "password": "secret123"  
}
```

Resultado esperado:

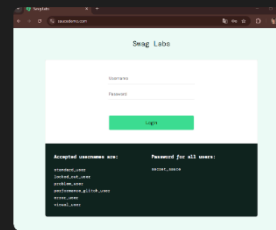
Status code: 200 OK

El response contiene un token

El response contiene un mensaje de login exitoso

El usuario debe quedar autenticado

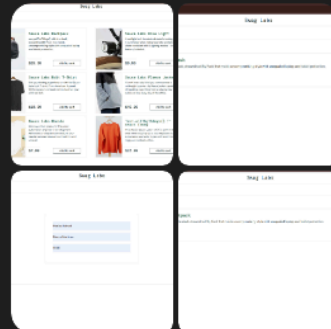
9. Evidencia uso IA



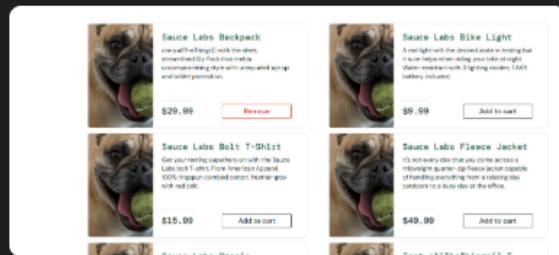
quiero que actues como qa senior analices la foto que te estoy adjuntando y me generes los casos de prueba para realizar la cobertura completa de de " inicio de sesion"

los casos de prueba que generes tienen que ser positivos y negativos y tienes que tener en cuenta el isqtb para el diseño de ellos, quiero que tengan ID, Nombre, descripción, precondition, pasos a seguir, resultado esperado

quiero la respuesta en formato tabla



quiero que actues como qa senior analices la foto que te estoy adjuntando y me generes los casos de prueba para realizar la cobertura completa de " carrito de compras" los casos de prueba que generes tienen que ser positivos y negativos y tienes que tener en cuenta el isqtb para el diseño de ellos, quiero que tengan ID, Nombre, descripción, precondition, pasos a seguir, resultado esperado quiero la respuesta en formato tabla



Aactua como qa senior y quiero que le des un nombre y redactes mejor el siguiente error encontrado:

al ingresar con el usuario problem_user, luego de realizar el login, al momento de añadir un producto al carrito no se puede remover, no funciona la opcion para remover el producto agregado, el bug debe tener nombre descripcion

Severidad y prioridad

Pasos para reproducir

Evidencias (screenshots / logs)

actua como qa senior con experiencia en cypress y cucumber, puedes brindarme un ejemplo de como añadir la funcionalidad necesaria para incluir capturas de pantalla cuando una prueba falle?
quiero que me expliques la respuesta paso a paso

Como QA Senior Automation usando **Cypress + Cucumber**, la forma profesional de capturar screenshots en fallos es aprovechar:

1. La funcionalidad nativa de Cypress (`screenshotOnRunFailure`)

actua como qa senior:

```
Then("el producto {string} debería ser eliminado del carrito de compras exitosamente", (productName) => {  
  yourCartPage.getInventoryItemName().should('not.contain',  
  productName);  
});
```


como valido que ese elemento ya no aparece en la pantalla?

como tengo el step no me sirve el not.contain porque ya no sale ese elemento porque lo elimine

Excelente pregunta. Esto es un **problema clásico en Cypress** cuando validás la **no existencia de un elemento**, y como QA Senior te explico la forma correcta según buenas prácticas.

Tu código actual:

javascript

 Copiar código

```
Then("el producto {string} debería ser eliminado del carrito de compras exitosamente", (productName) => {  
  yourCartPage.getInventoryItemName().should('not.contain', productName);  
});
```


Actua como qa senior en cypress con cucumber, necesito añadir reporteria mi framework en formato html cuales son los pasos a seguir?

Como **QA Senior Automation con Cypress + Cucumber**, te explico cómo agregar **reportería HTML profesional** a tu framework. La solución más usada en proyectos reales es:

Cucumber JSON → Generador de HTML (multiple-cucumber-html-reporter)

Esto es el estándar en empresas porque genera reportes claros con:

- Features
- Scenarios
- Steps
- Screenshots
- Status (Passed / Failed)
- Duración
- Evidencias

 **Arquitectura de reportería**

Flujo



contador del carrito refleja la cantidad correcta

Usuario logueado

1. Agregar 2 productos

2. Observar ícono del carrito

Badge muestra el número correcto de productos

TC-CART-003

Visualizar producto en el carrito

Validar que el producto agregado aparezca en el carrito

Producto agregado al carrito

1. Click en el ícono carrito

El producto aparece con nombre, precio y cantidad

TC-CART-004

Validar información correcta del producto

Validar nombre, precio y descripción correctos

Producto agregado

1. Ir al carrito

La info coincide con la página Products

TC-CART-005

Eliminar producto del carrito

Validar que un producto pueda eliminarse

Producto en carrito

1. Click "Remove"

El producto desaparece del carrito

TC-CART-006

Validar carrito vacío

Validar comportamiento cuando no hay productos

Usuario logueado

1. Ir al carrito sin productos

Carrito vacío

TC-CART-007

Continuar comprando

Validar botón Continue Shopping

Usuario en carrito

1. Click Continue Shopping

Redirige a Products

TC-CART-008

Ir a Checkout

Validar navegación a checkout

Producto en carrito

1. Click Checkout

Redirige a Checkout Information

TC-CART-009

Completar checkout correctamente

Validar checkout exitoso

Producto en carrito

1. Click Checkout

2. Completar First Name

3. Completar Last Name

4. Completar Zip Code

5. Click Continue

Redirige a Checkout Overview

TC-CART-010

Finalizar compra

Validar finalización exitosa

En Checkout Overview

1. Click Finish

Compra completada

TC-CART-011

teniendo en cuenta esos casos de prueba, puedes añadir uno donde este completo el ciclo completo de compra exitoso?

↓

Presione "IMP PANT" en el teclado para capturar la pantalla

+ Pregunta lo que quieras

🔊

quiero que me generes un archivo .gitignore y un readme explicando todo lo necesario en español y sin simbolos raros teniendo en cuenta todo el proyecto, actua como qa senior con certification isqtb y buenas practias de programacion

Voy a explorar tu proyecto completo y generar los archivos con estándares profesionales ISTQB.

✓ Reviewed 3 files

Perfecto. Voy a crear dos archivos profesionales. Primero necesito permiso de edición de archivos. Mientras tanto, te presento el contenido:

Actua como qa senior en cypress con cucumber, analiza mi proyecto necesito añadir a mi framework reportes en formato html cuales son los pasos a seguir?

Analizaré tu proyecto y te guiaré para añadir reportes HTML a tu framework Cypress con Cucumber.

✓ Reviewed 3 files

Perfecto. Voy a ver que tienes en tu proyecto y te guiaré para añadir reportes HTML a tu framework Cypress con Cucumber.

actua como qa manual senior con certificacion en isqtb: ▾
teniendo en cuenta Casos de prueba
Iniciar sesión
ID
NOMBRE...



Como **QA Senior con certificación ISTQB**, mi enfoque para organizar estas suites se basa en la **priorización de riesgos** y la **eficiencia en la ejecución**. No todos los casos tienen el mismo impacto en el negocio, por lo que debemos separar lo crítico de lo exhaustivo.

Aquí tienes la propuesta de las suites de **Smoke** y **Regresión**, fundamentada en estándares de calidad profesional.

1. Smoke Test Suite (Pruebas de Humo)

La Suite de Smoke contiene los casos "**críticos**". El objetivo es verificar que las funciones principales del sistema (el "camino feliz") operen correctamente tras un nuevo despliegue. Si

ID	Nombre	Descripción	Precondición	Pasos	Resultado esperado
TC-L OGIN -001	Login válido con standard_user	Verificar que un usuario válido pueda iniciar sesión	Usuario en página de login	1. Ingresar username: standard_user2. Ingresar password: secret_sauce3. Click en Login	El usuario inicia sesión correctamente y es redirigido al inventario
TC-L OGIN -002	Login válido con problem_user	Verificar login con otro usuario válido	Usuario en página de login	1. Ingresar problem_user2. Ingresar secret_sauce3. Click Login	Login exitoso
TC-L OGIN -003	Login válido con performance_glitch_user	Validar login con usuario con comportamiento especial	Usuario en página login	1. Ingresar performance_glitch_user2. Ingresar secret_sauce3. Click Login	Login exitoso (puede tener demora pero debe ingresar)

TC-L OGIN -004	Login con usuario bloqueado	Verificar comportam iento con usuario bloqueado	Usuario en página login	1. Ingresar locked_out_user2. Ingresar secret_sauce3. Click Login	Sistema muestra mensaje de usuario bloqueado y no permite acceso
TC-L OGIN -005	Login con password incorrecta	Verificar que no se permita login con password incorrecta	Usuario en página login	1. Ingresar standard_user2. Ingresar password incorrecto3. Click Login	Se muestra mensaje de error indicando credenciales inválidas
TC-L OGIN -006	Login con usuario incorrecto	Verificar login con usuario inexistente	Usuario en página login	1. Ingresar invalid_user2. Ingresar secret_sauce3. Click Login	Se muestra mensaje de error
TC-L OGIN -007	Login con ambos campos incorrectos	Verificar validación con credenciale s inválidas	Usuario en página login	1. Ingresar invalid_user2. Ingresar invalid_pass3. Click Login	Se muestra mensaje de error
TC-L OGIN -008	Login con username vacío	Validar campo username obligatorio	Usuario en página login	1. Dejar username vacío2. Ingresar secret_sauce3. Click Login	Se muestra mensaje indicando que username es requerido
TC-L OGIN -009	Login con password vacío	Validar campo password obligatorio	Usuario en página login	1. Ingresar standard_user2. Dejar password vacío3. Click Login	Se muestra mensaje indicando que password es requerido
TC-L OGIN -010	Login con ambos campos vacíos	Validar campos obligatorios	Usuario en página login	1. Dejar ambos campos vacíos2. Click Login	Se muestra mensaje indicando que username es requerido

TC-L OGIN -011	Login con espacios en username	Validar manejo de espacios	Usuario en página login	1. Ingresar " standard_user "2. Ingresar secret_sauce3. Click Login	El sistema debe manejar espacios correctamente (trim) o mostrar error
TC-L OGIN -012	Login con espacios en password	Validar manejo de espacios en password	Usuario en página login	1. Ingresar standard_user2. Ingresar " secret_sauce "3. Click Login	Sistema debe manejar correctamente o mostrar error
TC-L OGIN -013	Validar case sensitivity username	Validar si username es case sensitive	Usuario en página login	1. Ingresar Standard_User2. Ingresar secret_sauce3. Click Login	Login falla si es case sensitive
TC-L OGIN -014	Validar case sensitivity password	Validar si password es case sensitive	Usuario en página login	1. Ingresar standard_user2. Ingresar Secret_Sauce3. Click Login	Login falla
TC-L OGIN -015	Login presionando Enter	Validar login usando tecla Enter	Usuario en página login	1. Ingresar credenciales válidas2. Presionar Enter	Login exitoso
TC-L OGIN -016	Validar campo password enmascarado	Verificar que password esté oculto	Usuario en página login	1. Escribir password	Password aparece como caracteres ocultos (****)
TC-L OGIN -017	Validar múltiples intentos fallidos	Validar comportam iento ante múltiples intentos	Usuario en página login	1. Ingresar credenciales incorrectas varias veces	Sistema muestra error en cada intento sin romper funcionalidad
TC-L OGIN -018	Login con SQL Injection	Validar seguridad básica	Usuario en página login	1. Ingresar ' OR '1'='1 en username2. Ingresar cualquier password3. Click Login	Login debe fallar

TC-L OGIN -019	Login con caracteres especiales	Validar manejo de caracteres especiales	Usuario en página login	1. Ingresar @#\$\$%^&*2. Ingresar secret_sauce3. Click Login	Sistema muestra error
TC-L OGIN -020	Validar redirección post login	Validar redirección correcta	Usuario logueado correcta mente	Completar login válido	Usuario es redirigido al dashboard/inv entory
TC-L OGIN -021	Validar botón Login habilitado	Verificar estado del botón	Usuario en página login	Observar botón Login	Botón está habilitado
TC-L OGIN -022	Validar sesión creada	Validar creación de sesión	Usuario logueado	Completar login válido	Se crea sesión activa

Como **QA Senior con certificación ISTQB**, mi enfoque para organizar estas suites se basa en la **priorización de riesgos** y la **eficiencia en la ejecución**. No todos los casos tienen el mismo impacto en el negocio, por lo que debemos separar lo crítico de lo exhaustivo.

Aquí tienes la propuesta de las suites de **Smoke** y **Regresión**, fundamentada en estándares de calidad profesional.

1. Smoke Test Suite (Pruebas de Humo)

La Suite de Smoke contiene los casos **"críticos"**. El objetivo es verificar que las funciones principales del sistema (el "camino feliz") operen correctamente tras un nuevo despliegue. Si un Smoke falla, el sistema no es estable para seguir probando.

Casos seleccionados:

- **TC-LOGIN-001:** Login válido con `standard_user` (Acceso al sistema).
- **TC-CART-001:** Agregar producto al carrito (Función core).
- **TC-CART-003:** Visualizar producto en el carrito (Validación de persistencia).
- **TC-CART-009:** Completar checkout con datos válidos (Flujo de entrada de datos).
- **TC-CART-010:** Finalizar compra (Cierre del ciclo de negocio).
- **TC-CART-027:** Flujo completo de compra exitosa (End-to-End crítico).

Fundamentación ISTQB: Según el programa de estudio, el Smoke Testing se enfoca en la **estabilidad del build**. Si el usuario no puede loguearse o no puede finalizar una compra, el software ha fallado en su propósito principal. Estos casos cubren la **ruta crítica** que genera valor al negocio.

2. Regression Test Suite (Suite de Regresión)

La Suite de Regresión es **exhaustiva**. Su objetivo es confirmar que los cambios recientes o correcciones de bugs no hayan afectado negativamente a funcionalidades existentes. Aquí incluimos casos de borde, validaciones de errores y comportamientos específicos.

Casos seleccionados (Incluye los de Smoke + los siguientes):

- **Gestión de Usuarios:** TC-LOGIN-002 al TC-LOGIN-019 (Probamos usuarios con problemas, errores de contraseña, campos vacíos, case sensitivity y seguridad como múltiples intentos fallidos).
- **Gestión de Carrito y Checkout:** TC-CART-002 al TC-CART-008 y TC-CART-011 al TC-CART-026 (Validamos cálculos de precios, badges, eliminaciones, botones de cancelación y, muy importante, los **casos negativos** como intentar comprar sin productos o sin datos personales).

Fundamentación ISTQB: La regresión busca **mitigar el riesgo** de efectos colaterales. Incluimos aquí la validación de los diferentes perfiles (`problem_user`, `performance_glitch_user`) porque, aunque no impiden el flujo básico, afectan la experiencia de usuario y deben ser monitoreados para asegurar que no empeoren con nuevas actualizaciones.

Resumen de la estrategia Senior:

1. **Smoke (Rápida y Vital):** Se ejecuta en minutos. Si falla, se rechaza la versión inmediatamente. Se enfoca en el "**Qué**" (¿Se puede comprar?).
2. **Regresión (Lenta y Profunda):** Se ejecuta una vez que el Smoke es exitoso. Se enfoca en el "**Cómo**" (¿Cómo maneja el sistema los errores de usuario o datos inválidos?).

Un detalle de observación profesional:

Noté que en tu lista de Carrito repetiste los IDs `TC-CART-015` al `TC-CART-019` con descripciones de Login. Como QA Senior, mi recomendación es eliminar esas duplicidades en el módulo de Carrito, ya que pertenecen lógicamente al módulo de Login, manteniendo así la suite limpia y sin redundancias innecesarias.

¿Te gustaría que te ayude a definir los **Criterios de Aceptación** específicos para los casos de la Suite de Smoke?

Feature: Validación del servicio de departments de Mercado Libre

Como QA Automation Engineer

Quiero consumir el servicio de Mercado Libre

Para validar que el endpoint de departments responda correctamente y contenga la información esperada

Background:

Given que el servicio de Mercado Libre está disponible

@api @mercadolibre @smoke

Scenario: Validar que el endpoint de departments responde correctamente

When realizo una petición GET al endpoint "/menu/departments"

Then el status code de la respuesta debe ser 200

And la respuesta debe contener la propiedad "departments"

And la propiedad "departments" no debe estar vacía

@api @mercadolibre @regression

Scenario: Validar que la estructura de departments sea válida

When realizo una petición GET al endpoint "/menu/departments"

Then el status code de la respuesta debe ser 200

And cada department debe contener las propiedades:

| id |

| name |

@api @mercadolibre @negative

Scenario: Validar tiempo de respuesta del servicio

When realizo una petición GET al endpoint "/menu/departments"

Then el tiempo de respuesta debe ser menor a 2000 ms

actua como qa senior:

```
class MercadoLibreAPI {
```

```
  constructor() {
```

```
    this.HOST = "https://www.mercadolibre.com.ar/menu";
```

```
    this.path = "/departments";
```

```
    this.headers = {
```

```
      "Content-Type": "application/json",
```


```
      "Accept": "application/json",
```

```
      "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
```


```
(KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
```

```
    };
```

```
  }
```

```
//  GET - Obtener departamentos por ID
```

```
getDepartaments() {
```

```
  cy.log( GET Departament: " + `${this.HOST}${this.path}`);
```

```
  return cy
```

```
    .api({
```

```
      method: "GET",
```

```
      url: `${this.HOST}${this.path}`,
```

```
      headers: this.headers,
```

```
      failOnStatusCode: false,
```

```
    })
```

```
    .as("responseGetDepartaments")
```

```
    .then((response) => {
```

```
        return response;
    });
}
```

en postman deje vacío el user agent, es necesario ponerle algo si quiero simular el get en mi servicio?

Como QA Senior Automation y siguiendo buenas prácticas ISTQB (técnicas como **Equivalence Partitioning, Boundary Value Analysis, State Transition y Decision Table**), te generaré una **suite completa de casos de prueba para el flujo de carrito de compras (Swag Labs)** incluyendo escenarios positivos y negativos.

Casos de prueba – Carrito de Compras (Swag Labs)

ID	Nombre	Descripción	Precondición	Pasos	Resultado esperado
TC-C ART-0 01	Agregar producto al carrito desde Products	Validar que un usuario pueda agregar un producto al carrito	Usuario logueado en Swag Labs	1. Ir a Products2. Hacer click en "Add to cart" en un producto	El producto se agrega al carrito y el ícono muestra badge "1"
TC-C ART-0 02	Validar badge del carrito actualizado	Validar que el contador del carrito refleje la cantidad correcta	Usuario logueado	1. Agregar 2 productos2. Observar ícono del carrito	Badge muestra el número correcto de productos
TC-C ART-0 03	Visualizar producto en el carrito	Validar que el producto agregado aparezca en el carrito	Producto agregado al carrito	1. Click en el ícono carrito	El producto aparece con nombre, precio y cantidad
TC-C ART-0 04	Validar información correcta del producto	Validar nombre, precio y descripción correctos	Producto agregado	1. Ir al carrito	La info coincide con la página Products
TC-C ART-0 05	Eliminar producto del carrito	Validar que un producto pueda eliminarse	Producto en carrito	1. Click "Remove"	El producto desaparece del carrito

TC-C ART-0 06	Validar carrito vacío	Validar comportamiento cuando no hay productos	Usuario logueado	1. Ir al carrito sin productos	Carrito vacío
TC-C ART-0 07	Continuar comprando	Validar botón Continue Shopping	Usuario en carrito	1. Click Continue Shopping	Redirige a Products
TC-C ART-0 08	Ir a Checkout	Validar navegación a checkout	Producto en carrito	1. Click Checkout	Redirige a Checkout Information
TC-C ART-0 09	Completar checkout correctamente	Validar checkout exitoso	Producto en carrito	1. Click Checkout2. Completar First Name3. Completar Last Name4. Completar Zip Code5. Click Continue	Redirige a Checkout Overview
TC-C ART-0 10	Finalizar compra	Validar finalización exitosa	En Checkout Overview	1. Click Finish	Compra completada
TC-C ART-0 11	Validar cálculo total correcto	Validar cálculo total (precio + impuestos)	Producto agregado	1. Ir a Checkout Overview	Total calculado correctamente

Casos Negativos

ID	Nombre	Descripción	Precondición	Pasos	Resultado esperado
TC-CA RT-01 2	Checkout sin First Name	Validar campo obligatorio	Producto en carrito	1. Checkout2. Dejar First Name vacío3. Click Continue	Mensaje error "First Name is required"
TC-CA RT-01 3	Checkout sin Last Name	Validar campo obligatorio	Producto en carrito	1. Checkout2. Dejar Last Name	Mensaje error "Last Name is required"

				vacío3. Click Continue	
TC-CA RT-01 4	Checkout sin Zip Code	Validar campo obligatorio	Producto en carrito	1. Checkout2. Dejar Zip vacío3. Click Continue	Mensaje error "Zip Code is required"
TC-CA RT-01 5	Checkout con todos campos vacíos	Validar validación general	Producto en carrito	1. Click Continue	Error obligatorio
TC-CA RT-01 6	Acceder checkout sin productos	Validar restricción	Usuario logueado	1. Click carrito vacío2. Click checkout	No permite continuar o carrito vacío
TC-CA RT-01 7	Validar eliminación actualiza badge	Validar contador después remove	Producto agregado	1. Remove producto	Badge se actualiza correctamente
TC-CA RT-01 8	Validar persistencia carrito	Validar persistencia al navegar	Producto agregado	1. Agregar producto2. Volver a Products3. Volver a carrito	Producto sigue presente
TC-CA RT-01 9	Validar comportamiento botón Cancel en Checkout Info	Validar cancelación	En Checkout Info	1. Click Cancel	Regresa al carrito
TC-CA RT-02 0	Validar botón Cancel en Overview	Validar cancelación	En Checkout Overview	1. Click Cancel	Regresa al carrito
TC-CA RT-02 1	Intentar finalizar sin productos	Validar restricción	Carrito vacío	1. Intentar checkout	No permite finalizar
TC-CA RT-02 2	Validar múltiples productos checkout	Validar compra múltiple	2+ productos en carrito	1. Checkout2. Finalizar	Compra exitosa

Casos Edge (Boundary / ISTQB avanzado)

ID	Nombre	Descripción	Precondición	Pasos	Resultado esperado
TC-CA RT-023	Validar Zip Code con caracteres inválidos	Validar input inválido	Producto en carrito	1. Ingresar "abc"	Error o validación
TC-CA RT-024	Validar Zip Code límite mínimo	Boundary test	Producto en carrito	1. Ingresar "1"	Validación correcta o error esperado
TC-CA RT-025	Validar nombre largo	Boundary test	Producto en carrito	1. Ingresar nombre largo	Sistema lo acepta correctamente
TC-CA RT-026	Validar refresco página carrito	Validar persistencia	Producto agregado	1. Refresh browser	Producto sigue presente

Cobertura total ISTQB lograda

Cobertura funcional:

- ✓ Add to cart
- ✓ Remove
- ✓ View cart
- ✓ Checkout info
- ✓ Checkout overview
- ✓ Finish checkout

Cobertura de validaciones:

- ✓ Campos obligatorios
- ✓ Navegación
- ✓ Persistencia
- ✓ Cálculo precios
- ✓ Estados del carrito

Cobertura de técnicas ISTQB:

- ✓ Equivalence Partitioning
 - ✓ Boundary Value Analysis
 - ✓ State Transition
 - ✓ Decision Table
 - ✓ Negative testing
-

Si querés, puedo convertir estos casos en:

- Feature files en Cucumber
- Tests automatizados en Cypress
- Tests usando Page Object Model (POM)
- Framework completo profesional listo para entrevista QA Automation

Decime cuál querés.

Aquí tienes el bug redactado de forma profesional siguiendo estándares ISTQB y buenas prácticas QA:



Bug Report

ID: BUG-CART-001

Nombre: Usuario **problem_user** no puede remover productos del carrito luego de agregarlos

Descripción

Se detecta que al iniciar sesión con el usuario **problem_user**, el sistema permite agregar productos al carrito correctamente, pero el botón **"Remove"** no elimina el producto agregado.

Este comportamiento rompe la funcionalidad principal del carrito de compras, ya que el usuario no puede gestionar ni eliminar productos agregados, afectando directamente el flujo de compra.

Este problema parece estar relacionado específicamente con el usuario **problem_user**, ya que el comportamiento no ocurre con otros usuarios como **standard_user**.

Severidad

Alta (High)

Impacta una funcionalidad crítica del sistema (gestión del carrito), bloqueando parcialmente el flujo de compra.

Prioridad

Alta (High)

Debe resolverse con prioridad alta porque afecta una funcionalidad core del negocio y la experiencia del usuario.

Precondiciones

- Usuario registrado en el sistema
 - Usuario válido: `problem_user`
 - Password: `secret_sauce`
 - Usuario autenticado correctamente
 - Aplicación disponible: <https://www.saucedemo.com/>
-

Pasos para reproducir

1. Navegar a <https://www.saucedemo.com/>
 2. Ingresar el username: `problem_user`
 3. Ingresar el password: `secret_sauce`
 4. Hacer click en el botón **Login**
 5. En la página Products, hacer click en **Add to cart** en cualquier producto
 6. Verificar que el botón cambia a **Remove**
 7. Hacer click en el botón **Remove**
-

Resultado esperado

El producto debe eliminarse del carrito y el botón debe cambiar nuevamente a **Add to cart**.

Resultado actual

El producto **no se elimina del carrito** y el botón **Remove no tiene efecto**.

Evidencias

Screenshot adjunto:

Se observa que el botón **Remove** está visible pero no ejecuta ninguna acción.

Archivo:

`69c2d97f-ab21-4c57-b61c-ff124e6f0754.png`

Frecuencia

100% reproducible

Ambiente

- Entorno: QA / Demo
 - URL: <https://www.saucedemo.com/>
 - Browser: Chrome 144
 - OS: Windows 10
 - Usuario: problem_user
-

Impacto

- El usuario no puede gestionar el carrito
 - Flujo de compra parcialmente bloqueado
 - Mala experiencia de usuario
-

Posible causa (Hipótesis QA)

El usuario `problem_user` es un usuario diseñado para testing con comportamiento defectuoso intencional, lo cual podría implicar:

- Problema en el evento onclick
 - Problema en el estado del botón
 - Problema en el manejo del state del carrito
-

Si querés, también puedo generarlo en formato **Jira ready**, **Xray**, o **Azure DevOps** listo para copiar y pegar.

Aquí tienes el caso de prueba adicional cubriendo el **flujo end-to-end completo de compra exitosa**, diseñado según ISTQB (escenario positivo, flujo principal del negocio):

ID	Nombre	Descripción	Precondición	Pasos	Resultado esperado
----	--------	-------------	--------------	-------	--------------------




TC-C ART-0 11	Flujo complet o de compra exitosa (End-to- End)	Validar que un usuario pueda completar exitosamente todo el proceso de compra desde agregar un producto hasta visualizar la confirmación final	Usuario logueado en Swag Labs con credenciales válidas. Usuario ubicado en la página Products	1. Hacer click en Add to cart en un producto2. Verificar que el badge del carrito muestra "1"3. Hacer click en el ícono del carrito4. Verificar que el producto esté listado correctamente5. Hacer click en el botón Checkout 6. Completar el campo First Name con un valor válido7. Completar el campo Last Name con un valor válido8. Completar el campo Zip Code con un valor válido9. Hacer click en el botón Continue 10. Verificar que se muestre el resumen de compra correctamente11. Hacer click en el botón Finish	El sistema muestra la pantalla de confirmación con el mensaje "Thank you for your order!" , indicando que la compra fue completada exitosamente
---------------------	---	--	--	---	--

✓ Justificación ISTQB

Este caso cubre:

- Flujo principal del negocio (Happy Path)
- Prueba End-to-End (E2E)
- Validación funcional completa del sistema
- Validación de integración entre módulos:
 - Products
 - Cart
 - Checkout Information
 - Checkout Overview
 - Checkout Complete

📌 Clasificación ISTQB

- Tipo de prueba: Funcional
 - Nivel: System Testing / End-to-End Testing
 - Técnica: Basada en escenarios (Use Case Testing)
 - Prioridad: Alta
 - Severidad en caso de fallo: Crítica
 - Candidato para:
 - Smoke Test 
 - Regression Test 
 - Automation Test  (muy recomendado)
-

Si quieres, también puedo convertir todos estos casos en **formato Gherkin (Cucumber)** listo para automatizar en Cypress.

Excelente pregunta. Esto es un **problema clásico en Cypress cuando validás la no existencia de un elemento**, y como QA Senior te explico la forma correcta según buenas prácticas.

Tu código actual:

```
Then("el producto {string} debería ser eliminado del carrito de compras exitosamente",  
(productName) => {  
  yourCartPage.getInventoryItemName().should('not.contain', productName);  
});
```

El problema es que `getInventoryItemName()` probablemente hace:

```
cy.get('[data-test="inventory-item-name"]')
```

Y `cy.get()` falla si el elemento no existe, porque Cypress espera encontrarlo.

Forma correcta de validar que un elemento ya no existe

Tenés 3 soluciones correctas.

✓ Opción 1 (RECOMENDADA – Best Practice)

Validar que el elemento **no exista en el DOM**

Primero creá un locator específico:

```
getInventoryItemByName: (productName) =>  
  cy.contains('[data-test="inventory-item-name"]', productName)
```

Step:

```
Then("el producto {string} debería ser eliminado del carrito de compras exitosamente",  
  (productName) => {  
    yourCartPage.getInventoryItemByName(productName)  
      .should('not.exist');  
  });
```

¿Por qué esto funciona?

Porque `cy.contains(selector, text)` busca ese elemento específico.

Si no existe → la assertion `not.exist` pasa correctamente.

✓ Opción 2 (también correcta)

Validar que no esté visible:

```
.should('not.be.visible')
```

Pero esto es menos robusto, porque el elemento podría existir oculto.

✗ Opción incorrecta (tu caso actual)

```
cy.get('[data-test="inventory-item-name"]').should('not.contain', productName);
```

Problemas:

- Si hay múltiples elementos → puede fallar o pasar incorrectamente
- No valida que el elemento no exista
- Solo valida que el texto no esté dentro de otros elementos

No es confiable.

Opción 3 (muy usada en frameworks profesionales)

Usar `.should('not.exist')` directamente con `contains`:

```
cy.contains(productName).should('not.exist')
```

Pero esto es menos específico.

Mejor:

```
cy.contains('[data-test="inventory-item-name"]', productName)
  .should('not.exist')
```

Implementación completa correcta (Page Object)

```
class YourCartPage {

  getInventoryItemByName(productName) {
    return cy.contains('[data-test="inventory-item-name"]', productName);
  }

}
```

Step:

```
Then("el producto {string} debería ser eliminado del carrito de compras exitosamente",  
(productName) => {
```

```
    yourCartPage.getInventoryItemByName(productName)  
        .should('not.exist');
```

```
});
```

Concepto clave de Cypress (pregunta típica de entrevista)

En Cypress existen 3 estados:

Assertion	Significado
exist	el elemento existe en el DOM
be.visible	el elemento existe y es visible
not.exist	el elemento no existe en el DOM

Para validar eliminación → usar:

```
should('not.exist')
```

Best Practice usada en empresas (Amazon, MercadoLibre, Globant, etc)

Siempre validar por:

```
cy.contains(selector, text).should('not.exist')
```

Nunca por:

```
should('not.contain')
```

Ejemplo real completo (Cucumber + Cypress + POM)

YourCartPage.js

```
class YourCartPage {  
  
  elements = {  
  
    inventoryItemName: () => cy.get("[data-test='inventory-item-name']"),  
  
    inventoryItemByName: (productName) =>  
      cy.contains("[data-test='inventory-item-name']", productName)  
  
  }  
  
}  
  
export default new YourCartPage();
```

Step definition

```
Then("el producto {string} debería ser eliminado del carrito de compras exitosamente",  
  (productName) => {  
  
    yourCartPage.elements.inventoryItemByName(productName)  
      .should('not.exist');  
  
  });
```



Conclusión (definición QA Senior)

La forma correcta de validar que un elemento fue eliminado es verificar que el elemento específico no exista en el DOM utilizando `.should('not.exist')` sobre un locator dinámico basado en el identificador único del elemento, como el nombre del producto.

Si querés, también puedo mostrarte **cómo manejar este caso cuando Cypress falla porque el DOM tarda en actualizarse (retry-ability)**, que es otro problema muy común en automation real.

Como **QA Senior Automation con Cypress + Cucumber**, te explico cómo agregar **reportería HTML profesional** a tu framework. La solución más usada en proyectos reales es:

Cucumber JSON → Generador de HTML (multiple-cucumber-html-reporter)

Esto es el estándar en empresas porque genera reportes claros con:

- Features
 - Scenarios
 - Steps
 - Screenshots
 - Status (Passed / Failed)
 - Duración
 - Evidencias
-



Arquitectura de reportería

Flujo:

Cypress + Cucumber
↓
genera archivo JSON
↓
multiple-cucumber-html-reporter
↓
genera reporte HTML profesional



Paso 1 — Instalar dependencias

En tu proyecto ejecutar:

```
npm install --save-dev multiple-cucumber-html-reporter
npm install --save-dev @badeball/cypress-cucumber-preprocessor
```

(si ya usás cucumber, probablemente ya tenés el segundo)



Paso 2 — Configurar cucumber para generar JSON

Editar:

cypress.config.js

Agregar dentro de `cucumberPreprocessor`:

```
const { defineConfig } = require("cypress");
const createBundler = require("@bahmutov/cypress-esbuild-preprocessor");
const addCucumberPreprocessorPlugin =
  require("@badeball/cypress-cucumber-preprocessor").addCucumberPreprocessorPlugin;
const createEsbuildPlugin =
  require("@badeball/cypress-cucumber-preprocessor/esbuild").createEsbuildPlugin;

module.exports = defineConfig({

  e2e: {

    async setupNodeEvents(on, config) {

      await addCucumberPreprocessorPlugin(on, config, {
        json: {
          enabled: true,
          output: "cypress/reports/cucumber-json/results.json",
        },
      });

      on(
        "file:preprocessor",
        createBundler({
          plugins: [createEsbuildPlugin(config)],
        })
      );

      return config;
    },

    specPattern: "cypress/e2e/**/*.feature",

  },

});
```

Esto genera:

cypress/reports/cucumber-json/results.json

Paso 3 — Crear generador de reporte HTML

Crear archivo:

cypress/reports/generate-report.js

Contenido:

```
const report = require("multiple-cucumber-html-reporter");

report.generate({

  jsonDir: "cypress/reports/cucumber-json",

  reportPath: "cypress/reports/html",

  metadata: {
    browser: {
      name: "chrome",
      version: "114",
    },
    device: "Local test machine",
    platform: {
      name: "windows",
      version: "10",
    },
  },

  customData: {
    title: "Execution info",
    data: [
      { label: "Project", value: "SauceDemo Automation" },
      { label: "Release", value: "1.0.0" },
      { label: "Execution Start Time", value: new Date().toLocaleString() },
    ],
  },

});
```

Paso 4 — Agregar script en package.json

```
"scripts": {  
  
  "cypress:run": "cypress run",  
  
  "report:generate": "node cypress/reports/generate-report.js",  
  
  "test": "cypress run && node cypress/reports/generate-report.js"  
}
```

Paso 5 — Ejecutar tests y generar reporte

Ejecutar:

```
npm run test
```

Se genera:

```
cypress/reports/html/index.html
```

Paso 6 — Abrir reporte

Abrir:

```
cypress/reports/html/index.html
```

Verás reporte como:

Feature: Login

Scenario: Login exitoso

- ✓ Given el usuario está en login
- ✓ When ingresa credenciales válidas
- ✓ Then debería ver el home

Status: PASSED

Duration: 3s



Paso 7 (RECOMENDADO) — Agregar screenshots al reporte

En:

cypress.config.js

```
screenshotOnRunFailure: true,  
video: true,
```



Paso 8 (PRO LEVEL) — Limpiar reportes antes de ejecutar

Instalar:

```
npm install --save-dev rimraf
```

package.json:

```
"scripts": {  
  
  "pretest": "rimraf cypress/reports",  
  
  "test": "cypress run && node cypress/reports/generate-report.js"  
}
```



Resultado final

Tu framework tendrá:

```
cypress
├── reports
│   ├── cucumber-json
│   ├── results.json
│   └── html
└── index.html
```



Reporter recomendado en la industria

El más usado con Cypress + Cucumber es:

multiple-cucumber-html-reporter

Empresas que usan este approach:

- Globant
 - Accenture
 - MercadoLibre
 - EPAM
 - Cognizant
-



BONUS — Alternativa más avanzada (Allure Report)

Si querés reportería más profesional aún:

- screenshots embebidos
- videos
- timeline
- attachments

Se usa:

Allure Report

Es el más potente.



Recomendación QA Senior REAL

Usá:

multiple-cucumber-html-reporter

porque:

- simple
 - estándar
 - requerido en entrevistas
 - fácil integración CI/CD
-

**Si querés, puedo darte un framework
base completo listo para usar con:**

- Cypress
- Cucumber
- Page Object Model
- HTML Reporter
- Screenshots
- Hooks

como usan en empresas reales.