

# PLAN DE PRUEBAS

# SAUCEDEMO

# Versión 1.0

## CONTROL DE CAMBIOS AL PLAN

[illegible]


CONTROL DE CAMBIOS TEMPLATE

VERSIÓN	FECHA	CAMBIOS RESPECTO DE LA VERSIÓN ANTERIOR	PREPARADO POR	APROBADO POR
1.0	Febrero 2026	Versión Inicial	Nombre QA	Nombre QA Lead
			Adrian Spinosi	

## Tabla de Contenidos

<b>1. INTRODUCCIÓN</b>	<b>4</b>
1.1. OBJETIVO DEL PLAN DE PRUEBAS	4
1.2. DOCUMENTOS RELACIONADOS	4
<b>2. ALCANCE DE LAS PRUEBAS</b>	<b>5</b>
2.1. REQUERIMIENTOS DE PRUEBAS INCLUIDOS	5
2.2. REQUERIMIENTOS DE PRUEBAS EXCLUIDOS	6
2.3. CASOS DE PRUEBAS INCLUIDOS	7
2.4. CASOS DE PRUEBAS EXCLUIDOS	8
2.5. ENTORNO DE PRUEBAS	9
<b>3. ESTRATEGIA DE LAS PRUEBAS</b>	<b>10</b>
3.1. CONFIGURACION DE LAS PRUEBAS	10
3.2. ORDEN DE EJECUCION DE LAS PRUEBAS	11
3.3. CRITERIOS DE INICIO Y TERMINOS DE LAS PRUEBAS	13
3.4. EQUIPOS DE PRUEBAS Y RESPONSABILIDADES	14
3.5. GESTION DE INCIDENCIAS	15
<b>4. ANEXOS</b>	<b>16</b>
4.1. ENTREGABLES	16

## 1. INTRODUCCIÓN

### 1.1. OBJETIVO DEL PLAN DE PRUEBAS

*El plan de pruebas tiene como objetivo identificar claramente cuál es el alcance respecto a lo que se incluye y excluye de las actividades de testeo, el orden de ejecución de las pruebas, los recursos necesarios tanto de data como de ambientes o infraestructura, los criterios de inicio y término, participantes del proyecto y todo lo relevante para llevar a cabo la actividad y finalmente aprobar el “paso a producción” de la solución, para así asegurarnos de satisfacer los requerimientos definidos por los usuarios de las distintas áreas*

### 1.2. DOCUMENTOS RELACIONADOS

NOMBRE	DESCRIPCIÓN
Repositorio del proyecto	<a href="#">Link</a>
Issue tracker	<a href="#">Link</a>
Casos de prueba	<a href="#">Link</a>

## 2. ALCANCE DE LAS PRUEBAS

### 2.1. REQUERIMIENTOS DE PRUEBAS INCLUIDOS

*El alcance de las pruebas a realizar es en base funcional con casos de prueba de ejecución manual y una suite de automatización con el framework Cypress lenguaje de programación Javascript con un apartado de pruebas de API dentro del mismo. (Alcance/Scope del testing)*

*Funcionalidades a cubrir Login y Carrito de la pagina => <https://www.saucedemo.com/>*

### 2.2. REQUERIMIENTOS DE PRUEBAS EXCLUIDOS

REQUERIMIENTOS	MOTIVO
Performance	Fuera del scope
Seguridad	Fuera del scope
Funcional	Caja negra, Smoke Test, Regresión.

### 2.3.CASOS DE PRUEBAS INCLUIDOS

ID	TÍTULO	ESTADO
TC-1	Login con credenciales válidas	PASS
TC-2	Login con credenciales incorrectas	PASS
TC-3	Login con username valido y pass incorrecta	PASS
TC-4	Login con username valido y pass vacia	PASS
TC-5	Login con username y pass valido	PASS
TC-6	Login con pass válido y username vacío	PASS
TC-7	Carrito de compras - Agregar 1 producto al carrito	PASS
TC-8	Carrito de compras - Remover producto del inventario	PASS
TC-9	Carrito de compra - Remover producto desde el carrito	PASS
TC-10	Carrito de compras - Cancelar checkout	PASS
TC-11	Carrito de compras - Continuar comprando	PASS
TC-12	Carrito de compras - Finalizar checkout exitoso	PASS

## 2.4. ENTORNO DE PRUEBAS

*Las pruebas serán realizadas en un ambiente de QA*

*QA: <https://www.saucedemo.com/>*

## 3. ESTRATEGIA DE LAS PRUEBAS

### 3.1. CONFIGURACION DE LAS PRUEBAS

1	Data Pre-requisito	<ul style="list-style-type: none"><li>- Ambiente de pruebas</li><li>- Credenciales de ingreso</li></ul>
2	Modalidad de ejecución de Pruebas	<i>Se realizarán casos de prueba en QMETRY (Complemento de Jira), los cuales contemplarán las funcionalidades actuales del sistema. Incluye framework de automatización Cypress con Javascript y reporte Una vez realizado, se ejecutarán en el ambiente de testing.</i>
3	Tipos de Pruebas	<i>Se realizarán pruebas funcionales manuales, ejecutando casos de prueba y se complementa con pruebas automatizadas.</i>
4	Integración con otros sistemas	<i>No</i>
5	Pruebas de Regresión	<i>La prueba de regresión se correrá al finalizar el testeo y luego de la salida a producción para garantizar de que el trabajo quedó correctamente realizado</i>
6	Herramienta de Registro y Control de Pruebas	<i>Se utilizará QMETRY, plugin incorporado en el Jira del proyecto.</i>

7	Herramienta de Gestión de Incidencias	<i>Se utilizará Jira como issue tracker y paquete office para la evidencia de la ejecución con reporte de bugs manuales.</i>
---	---------------------------------------	--

### 3.2. ORDEN DE EJECUCIÓN DE LAS PRUEBAS

A continuación se despliega un calendario simple con la programación propuesta para la actividad de ejecución del Testing, de acuerdo a la estimación preliminar realizada:

#### 3.2.1 Cronograma de Actividades

Mes	Enero			<div>Febrero</div>				Marzo				Abril				Fechas Estimadas
	S 2	S 3	S 4	S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4	
Fases																
Análisis de testing																
Creación de casos de prueba																
Ejecución de las pruebas																
Paso a Producción																
Regresión																

### 3.3.CRITERIOS DE INICIO Y TÉRMINOS DE LAS PRUEBAS

#### Criterios de Inicio

Para comenzar la ejecución del Testing, se establece como hitos obligatorios la ejecución sin observaciones o errores de los siguientes pasos:

- Ambiente de prueba
- Plan de prueba actualizado y armado
- Documentación de las APIs
- Los TestCase deben estar diseñados y contruidos en un 100% de acuerdo a lo planificado.
- Datos de acceso al sistema alojado en el ambiente de prueba

#### Criterios de Término

- La cobertura de ejecución de las pruebas debe ser de un 100% (pruebas pasada exitosamente) de acuerdo a lo definido y planificado.
- No deben existir defectos con severidad crítica (nivel 1), al término de la etapa.
- Si por algún motivo externo o que escapa a lo definido en este plan, se deberá dejar evidencia por escrito del motivo por el cual se finaliza la actividad, con los correspondientes responsables de todas las partes.

3.4.EQUIPOS DE PRUEBAS Y RESPONSABILIDADES

NOMBRE	ROL	RESPONSABILIDAD
Spinosi, Adrian	QA Automation	Creación y ejecución de casos de prueba
Nenadovit, Emmanuel Angel	QA Lead	Supervisar planes y casos de prueba

### 3.5. GESTIÓN DE INCIDENCIAS

El proceso de gestión de incidentes utilizado corresponde al proyecto, el cual es Jira.

El proceso o Flujo de Incidencia se resumen de la siguiente forma:

1. Equipo de QA genera/reporta la incidencia (con evidencia adjunta), frente a un error en el caso de Prueba ejecutado, éste queda en *Jira* ejecutado y Fallado.
2. La incidencia es derivada desarrollador correspondiente, quién valida si efectivamente es o no incidencia:
  - a. No es Incidencia: Se rechaza el ticket.
  - b. Es incidencia, desarrollo corrige y posteriormente informar por *Jira* al equipo de QA, para su re-testeo, previa habilitación (de la solución) en el ambiente de pruebas.
3. Equipo de QA re-testea o valida nuevamente:
  - a. Sí validación resulta exitosa, Caso de Prueba queda Pasado OK en *X-RAY*, con evidencia respectiva, y la incidencia reportada queda en estado Cerrada, todo lo anterior en *X-RAY*
  - b. Si el error persiste, el equipo de QA Rechaza a Desarrollo la incidencia. El Error reportado queda en estado “Rechazado a Desarrollo”, a la espera que el desarrollador corrija.
4. Ante el punto anterior, el flujo se repite (desde el punto 2) y finaliza cuando la incidencia es corregida Finalmente.

### 3.6. PLANTILLA REPORTE DE BUGS

**ID:** El ID de cada BUG debe ser único para poder identificar esa incidencia (Lo pone automáticamente el issue tracker)

**Título:** [Pantalla] Título descriptivo del error

**Descripción:** Comentar brevemente de que se trata el error y que se logre entender cuál es la falla

**Precondiciones:** Que es lo que necesito tener previamente configurado para poder ejecutar los pasos.

#### Test Data

Usuario: admin

Password: 123456

#### Pasos de reproducción:

1- Paso 1

2- Paso 2

3- Paso 3

4- Paso 4

**Resultado Actual:** Que es lo que está pasando ahora

**Resultado Esperado:** Como debería funcionar la aplicación

**Screenshot/Video:** Captura de pantalla o video del error

### Otros campos a completar:

- Asignar siempre a un desarrollador o encargado
- Colocar siempre una criticidad/severidad
- Especificar en que sprint se va a arreglar, de lo contrario, colocarlo en el backlog
- Asociarlo a un test case

(Estos campos no van escritos dentro de la descripción, sino que son campos que trae el issue tracker)

### Campos Opcionales:

**Versión:** Versión de la aplicación (Mobile)

**Browser:** Especificar en que navegador se encontró la falla

**Ambiente:** Especificar en que ambiente se encontró el error

Ciclo de vida de un bug:



#### 4. ANEXOS

### Regresión

ID	TÍTULO	ESTADO
TC-2	Login con credenciales incorrectas	PASS
TC-3	Login con username valido y pass incorrecta	PASS
TC-4	Login con username valido y pass vacia	PASS
TC-5	Login con username y pass valido	PASS
TC-6	Login con pass válido y username vacío	PASS
TC-7	Carrito de compras - Agregar 1 producto al carrito	PASS
TC-8	Carrito de compras - Remover producto del inventario	PASS
TC-9	Carrito de compra - Remover producto desde el carrito	PASS
TC-10	Carrito de compras - Cancelar checkout	PASS
TC-11	Carrito de compras - Continuar comprando	PASS

### SMOKE

ID	TÍTULO	ESTADO
TC-1	Login con credenciales válidas	PASS
TC-12	Carrito de compras - Finalizar checkout exitoso	PASS

## 5. Plan de test de API

**1-Alcance y objetivos:** Evaluar que APIs y endpoints(GET, POST,PUT, DELETE) se van a probar y hacer foco en las funcionalidades criticas

**2-Entornos de prueba:** Configuración de ambientes a probar (QA, Dev, UAT)

**3-Herramientas:** Postman, Jmeter

**4-Pruebas funcionales:** Validar que la API cumple con los requisitos de la documentación como el tipo de dato y estructura del body, headers. Validar en el caso de que corresponda el impacto de la request en la base de datos.

**5-Pruebas negativas:** Verificar la gestión de errores y mensajes del mismo(ejemplo 400 bad request 404 not found)

**6-Prueba de rendimiento:** evaluar el consumo de la api con datos consecutivos o masivos para evaluar el tiempo de respuesta, escalabilidad y carga.

## **7- Validaciones:**

a- Código de estado HTTP

b- Carga de respuestas(JSON/XML) y validación de esquemas

c- Encabezados de respuesta(headers)

**8- Herramientas para Automatizaciones de API:** Postman(Runner), RestAssured, Karate DSL Framework, Pytest. Evaluar la cantidad de iteraciones y delay de acuerdo al tiempo de respuesta del ambiente en el que se está probando.

## **9-Ejemplo de reporte de evidencia manual para Pruebas de API**

# **INTEGRACIÓN - Rickandmarty - Character**

**CP001 - Rickandmarty - Endpoint /Character - Consulta de personaje no existente**

- **Indices**

<a href="#">1</a>	<a href="#">INDICES</a>	<a href="#">2</a>
<a href="#">2</a>	<a href="#">DATOS DE MAPEO</a>	<a href="#">3</a>
<a href="#">3</a>	<a href="#">PASOS</a>	<a href="#">3</a>
<a href="#">4</a>	<a href="#">RESULTADO ESPERADO-OBTENIDO</a>	<a href="#">4</a>

- **Datos de Mapeo**

ID: Valores informados en imagen adjunta

- **Pasos**

Given que quiero consumir la API de RickandMorty

When invoco el endpoint GET <https://rickandmortyapi.com/api/character/23213123>

Then recibiré un request con código de estado 404 Not Found por parte del sistema

And Se valida el mensaje de "error": "Character not found"

- **Resultado Esperado-Obtenido**

Resultado Obtenido: Postman debe dar un código 404 Not Found, y se debe visualizar un mensaje de "error": "Character not found"

Resultado Obtenido: Postman da un código 404 Not Found, y se visualiza un mensaje de "error": "Character not found"

GET https://rickandmortyapi.com/api/character/23213123

Send

Docs Params Auth Headers (6) Body Scripts Settings

Cookies

### Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

404 Not Found

295 ms 812 B

Save Response

{ } JSON Preview Debug with AI

```

1 {
2   "error": "Character not found"
3 }
```