

# Prueba de validación técnica

## Teoría

### Contexto general

La agencia está desarrollando un **dashboard de métricas digitales** que permite a los usuarios autenticados visualizar resultados de campañas, filtrar información por fecha o categoría, y exportar reportes.

El equipo está finalizando una nueva versión del módulo de filtrado, y antes de hacer merge a `develop` requiere validar **flujo completo de login → carga de métricas → aplicación de filtros → validación de resultados**.

El objetivo de esta prueba es **evaluar tu capacidad para diseñar, estructurar y pensar como QA de producto digital**, combinando razonamiento funcional, automatización y control de calidad en entornos CI/CD.

### Objetivo de la prueba

Diseñar una propuesta de validación de calidad para este escenario, abarcando:

1. Estrategia general de pruebas.
2. Diseño de casos de prueba representativos.
3. Propuesta de automatización.

### Instrucciones generales

- El ejercicio debe ser **escrito** (no requiere código ejecutable).
- Puedes usar pseudocódigo, estructura YAML, o ejemplos sintéticos.
- Enfócate en demostrar **criterio técnico, claridad, y priorización de pruebas**.

#### 1. Estrategia general de pruebas (máx. 1 página)

Describe cómo abordarías la validación del flujo descrito. Incluye:

- Tipos de pruebas que considerarías, metodología que usarías para su redacción y aplicabilidad al caso señalado.
- Cómo estructurarías los ambientes de prueba.
- Qué aspectos validarías antes del merge.
- Cómo manejarías evidencias, reportes y comunicación de resultados.

#### 2. Casos de prueba representativos

Define los casos de prueba representativos, usa la estructura que consideres oportuna para claridad, contextualización y entendimiento de los casos diseñados y su finalidad.

#### 3. Propuesta de automatización (flujo crítico)

Elige uno de los casos anteriores

Explica:

- Qué herramienta/framework usarías tanto para automatización como para reportes, justifica la elección.
- Cómo estructurarías la prueba para el caso concreto.

# Práctica

## Prueba de automatización UI + API

La siguiente prueba tiene como objetivo evaluar tus habilidades prácticas en automatización de pruebas, tu capacidad para analizar flujos funcionales, estructurar casos de prueba y desarrollar scripts automatizados con buenas prácticas.

Deberás realizar automatizaciones sobre el sitio <https://automationexercise.com/>.

Este sitio web de e-commerce libre de producción permite realizar pruebas tanto de interfaz de usuario (UI) como de API, y dispone de endpoints documentados para testing.

## Objetivo general

Automatizar casos de prueba UI y API, demostrar que puedes integrar ambos dominios (front end y back end) en una estrategia de automatización, presentar un código limpio, bien estructurado y documentado.

## Entregables esperados

- Repositorio público en GitHub o archivo ZIP con la estructura de tu proyecto de automatización.
  - Recomendación de arquitectura:** Se valorará positivamente si el proyecto implementa una arquitectura modular o una estructura equivalente que permita mantener el código limpio, reutilizable y escalable.
- README explicativo.
- Scripts de automatización que cubran los casos solicitados (UI + API).

### 1.1 Casos de prueba UI

	≡ Flujo	≡ Objetivo	≡ Casos
1	Inicio de sesión	Validar autenticación positiva y negativa.	<div>1. Login exitoso<ul style="list-style-type: none"><li>Navegar a <i>Signup/Login</i></li><li>Ingresar correo y contraseña válidos</li><li>Validar saludo "Logged in as {username}"</li></ul></div> <div>2. Login fallido por contraseña incorrecta</div> <div>3. Login fallido por usuario inexistente</div> <div>4. Verificar que los mensajes de error aparezcan correctamente</div>
2	Opcional: Data-driven testing	<div>Agregar una prueba de login negativo basada en datos externos (CSV, JSON o una lista parametrizada).</div> <div>La prueba debe ejecutar múltiples combinaciones de usuarios/contraseñas inválidas y validar los mensajes de error correspondientes.</div>	<ul style="list-style-type: none"><li>usuario válido + contraseña inválida</li><li>correo con formato inválido</li><li>usuario inexistente</li><li>ambos campos vacíos</li></ul>
3	Agregar productos al carrito	Validar interacción y actualización de UI.	<div>1. Seleccionar un producto y hacer clic en <b>Add to cart</b></div> <div>2. Validar el modal de confirmación</div> <div>3. Navegar al carrito y verificar:<ul style="list-style-type: none"><li>Nombre del producto</li><li>Cantidad</li><li>Precio</li></ul></div> <div>4. Actualizar la cantidad del producto</div> <div>5. Eliminar el producto del carrito</div> <div>6. Validar que el carrito quede vacío</div>
4	Checkout (sin pago real)	Comprobar flujo completo hasta página final.	<div>1. Desde carrito → Proceed to Checkout</div> <div>2. Validar dirección del usuario cargada</div> <div>3. Añadir comentario</div> <div>4. Hacer clic en <b>Place Order</b> (no procesa pago real)</div> <div>5. Validar mensaje o navegación final</div>

5	Logout	Validar cierre de sesión	<div>1. Clic en <b>Logout</b></div> <div>2. Validar redirección a página de Login</div> <div>3. Intentar acceder a MyAccount sin sesión → debería bloquear o pedir login</div>
---	--------	--------------------------	--

## 1.2 Casos de prueba API

Mediante la API pública que tiene el sitio: [https://automationexercise.com/api\\_list](https://automationexercise.com/api_list)

	≡ Flujo	≡ Endpoint	≡ Casos
1	Login API	<div>POST /login</div>	<div>1. Login exitoso → validar <div>responseCode = 200</div></div> <div>2. Login con password inválido → <div>responseCode = 404</div></div> <div>3. Login con email inexistente → <div>responseCode = 404</div></div> <div>4. Validar estructura del JSON (mensaje, código, estado)</div>
2	Productos	<div>GET /productsList</div>	<div>1. Validar que el endpoint responde 200</div> <div>2. Validar que existan productos en el array</div> <div>3. Verificar estructura de un producto (id, name, price)</div> <div>4. Validar que los IDs no sean duplicados</div>
3	Crear usuario	<div>POST /createAccount</div>	<div>1. Creación exitosa con email nuevo</div> <div>2. Intento de creación con email ya existente → debe fallar</div> <div>3. Validar estructura de campos requeridos</div>
4	Validación cruzada UI + API		<div>• Hacer login por UI</div> <div>• Capturar nombre del usuario en UI</div> <div>• Validar contra valor retornado por <div>/getUserDetailByEmail</div></div>
5	Casos API	<div>Agregar al menos un caso adicional enfocado en escenarios negativos o de borde.</div> <div>Este punto permite evaluar tu capacidad de anticipar errores y validar robustez del backend.</div>	<div>• Validar comportamiento enviando parámetros vacíos o incorrectos.</div> <div>• Envío de request sin los headers requeridos (si aplica).</div> <div>• Validar estructura del error retornado por la API.</div> <div>• Validar tiempos de respuesta cuando corresponda (ej. asegurar que responde &lt; X ms).</div>