27 de octubre, 2025

CHALLENGE QA AUTOMATION

Presentado a
Mercap

Presentado por
Belardinelli Beker, Marianela

CONTENIDO

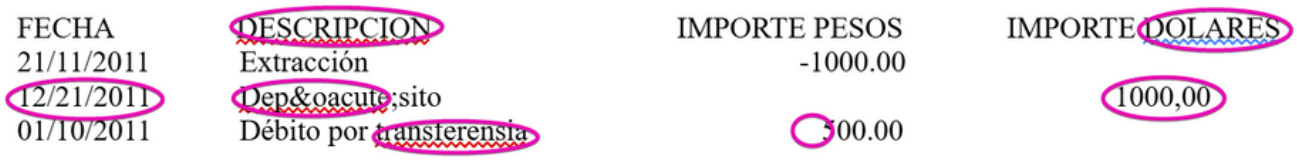
Resumen de entrega.

Se entrega:

- Este documento PDF con la resolución de los ejercicios 1 al 8
- Carpeta “Sauce Demo - Cypress Tests” en repositorio:
  🌐 GitHub - MarBelardineli/Mercap-challenge-QA: Challenge QA Automation
  con la resolución del ejercicio 9.
- Dos archivos formato JSON correspondientes a la resolución del ejercicio 10.

Ejercicio 1:

Detectar, marcar con un círculo, y explicar brevemente los defectos, si es que hubiere, en el siguiente ejemplo de una consulta de movimientos de una cuenta (tener en cuenta la unificación de criterio y que sólo se detallan registros de movimientos, sin importar el saldo de la cuenta):



FECHA	DESCRIPCION	IMPORTE PESOS	IMPORTE DOLARES
21/11/2011	Extracción	-1000.00	
12/21/2011	Depósito		1000,00
01/10/2011	Débito por transferensia	500.00	

Defectos detectados en la consulta de movimientos de cuenta:

1. **Formato de fecha inválido:**
    - Las fechas no siguen un formato unificado. Se debe utilizar DD/MM/AAAA (o, si así estuviese especificado, MM/DD/AAAA) de manera consistente.
    - Las fechas no respetan un orden lógico ascendente o descendente.
2. **Errores de ortografía y acentuación:**
    - “Depósito” debería escribirse correctamente como “Depósito”.
    - La palabra “Transferencia” (o “*Transferensia*”) contiene errores ortográficos.
    - Los títulos de columnas como “DESCRIPCIÓN” y “DÓLARES” deben llevar tilde aunque estén en mayúscula.
3. **Errores de cálculo o visualización:**
    - El importe del débito por transferencia no está siendo restado correctamente.
    - El importe en dólares no está alineado de la misma forma que el importe en pesos (uno está alineado a la derecha y el otro centrado).

Ejercicio 2:

Detallar los campos que se deberían utilizar al informar un defecto y realice el reporte de todos los errores encontrados en el primer enunciado

ID	DESCRIPCIÓN	PASOS PARA REPRODUCIR	RESULTADO ESPERADO	RESULTADO ACTUAL	SEVERIDAD	PRIORIDAD	EVIDENCIA
BUG-001	Las fechas no siguen un formato unificado	1. Consultar movimientos de cuenta. 2. Visualizar columna de fechas	Todas las fechas se muestran con un mismo formato (DD/MM/AAAA o bien MM/DD/AAAA según especificación)	Las fechas se muestran con distintos formatos: DD/MM/AAAA y MM/DD/AAAA de forma indistinta	Alta	Alta	https://drive.google.
	El monto de débito por transferencia está siendo sumado en lugar de restado	1. Registrar o visualizar un movimiento de tipo “Débito”. 2. Comprobar el efecto sobre el total de la cuenta.	Los montos de débito deben restarse del saldo total.	El movimiento de $500 en débito se suma en lugar de restarse.	Alta	Alta	https://drive.google.
BUG-002	Las fechas no respetan un orden lógico	1. Consultar movimientos de cuenta. 2. Visualizar columna de fechas	Las fechas se muestran en orden descendente (o ascendente, según especificación)	Las fechas no siguen un orden cronológico	Media	Media	https://drive.google.
BUG-003	La tilde en “Depósito” se visualiza como entidad HTML	1. Consultar movimientos de cuenta. 2. Visualizar movimiento “Depósito” en la columna “Descripción”.	El texto debe mostrarse correctamente con tildes visibles.	La palabra “Depósito” se muestra como “Depósito”.	Media	Media	https://drive.google.
BUG-004	Error ortográfico en la palabra “Transferencia”	1. Consultar movimientos de cuenta 2. Observar el texto del movimiento de tipo “Transferencia”.	El texto debe mostrarse correctamente escrito como “Transferencia”.	Se visualiza “Transferensia” con ‘s’.	Media	Media	https://drive.google.
BUG-005	Tildes faltantes en los títulos “DESCRIPCIÓN” y “DÓLARES”	1. Ingresar a pantalla de consulta de movimientos de cuenta 2. Visualizar encabezados de las columnas	Los títulos deben llevar tilde aunque estén en mayúscula.	Los títulos en mayúscula no muestran tildes.	Baja	Baja	https://drive.google.
BUG-006	Los importes en dólares y pesos no están alineados de la misma forma	1. Ingresar a la consulta de movimientos. 2. Visualizar la columna de importes en ambas monedas.	Los importes deben estar alineados de forma uniforme (por ejemplo, todos a la derecha).	El importe en pesos está alineado a la derecha, mientras que el importe en dólares está centrado.	Baja	Media	https://drive.google.

Ejercicio 3:

Defina lo que son las siguientes técnicas de test y diseñe un caso de prueba para cada una (puede armar un escenario de ejemplo para utilizarlo para explicar cada técnica):

- **Pruebas funcionales**

Las pruebas funcionales se centran en **verificar que el sistema cumpla correctamente con los requisitos definidos**, validando que cada funcionalidad entregue los resultados esperados ante diferentes entradas o acciones del usuario.

Se evalúa la lógica del negocio, los flujos principales y alternativos, y la correcta integración entre los distintos componentes del sistema.

Ejemplo:

Registrar un movimiento de tipo Débito y verificar que:

- El sistema descuente correctamente el monto ingresado del saldo total de la cuenta.
- El registro se almacene y visualice en la lista de movimientos.
- Se muestre el tipo de operación y la fecha con el formato esperado.

Objetivo: asegurar que la funcionalidad cumpla con el comportamiento definido en los requisitos y no se produzcan errores de lógica o validación.

- **Pruebas no funcionales**

Se centran en evaluar **atributos de calidad del sistema**, más allá de su comportamiento funcional. Incluyen aspectos como rendimiento, seguridad, compatibilidad, usabilidad y estabilidad.

Su objetivo es garantizar que la aplicación sea eficiente, segura y ofrezca una buena experiencia de usuario.

Ejemplo:

Medir el tiempo de respuesta del sistema al registrar un movimiento y verificar que no supere los 3 segundos establecidos como criterio de rendimiento.

Objetivo: validar que el sistema mantenga un rendimiento adecuado y una experiencia de usuario fluida, incluso bajo condiciones normales de carga.

- **Pruebas de carga y estrés**

Estas pruebas analizan el **comportamiento y estabilidad del sistema ante altos volúmenes de operaciones o usuarios concurrentes.** Las pruebas de carga determinan el rendimiento bajo condiciones esperadas, mientras que las de estrés buscan el punto de falla del sistema al sobrepasar su capacidad.

Ejemplo:

Simular el registro simultáneo de 1000 movimientos de cuenta para evaluar:

- El tiempo de procesamiento de cada transacción.

- El uso de recursos del servidor (CPU, memoria, red).
- La capacidad del sistema para mantener la integridad de los datos sin caídas ni pérdidas de información.

Objetivo: identificar posibles cuellos de botella, degradaciones de rendimiento o fallas del sistema cuando se enfrenta a una carga alta o extrema.

Ejercicio 4:

Se necesita realizar un cálculo de bonos para los empleados. No puede ser negativo, pero puede ser cero. El bono está basado en la duración del empleo. Una persona puede ser empleado por menos o igual a 2 años, más que 2 años pero menos de 5 años, 5 a 10 años, o más de 10 años. Dependiendo del período del empleo, un empleado podrá obtener: ningún bono o un bono del 10%, 25% o 35%. ¿Cuántas particiones equivalentes se necesitan para testear el cálculo de bonos? Justifique su respuesta.

Se necesitan 4 particiones, de forma que cada partición represente un rango con un porcentaje distinto de bono:

i. 0-1 años (bono 0%)
ii. 2-5 años (bono 10%)
iii. 5 -10 años (bono 25%)
iv. + 10 años (bono 35%)

Ejercicio 5:

Los siguientes tres tests han sido ejecutados para el siguiente diagrama de flujo detallado debajo.
El test A cubre los caminos: A, B, D, E, G.
El test B cubre los caminos: A, B, D, E, F, G.
El test C cubre los caminos: A, C, F, C, F, C, F, G.
¿Cuál de las siguientes afirmaciones es correcta, en relación a la cobertura de los flujos de decisión?:

La decisión F no ha sido testeada completamente, ya que ningun test cubre el camino D → F

Ejercicio 6:

¿En qué te basarías para estimar el esfuerzo que te llevará crear y ejecutar los casos de prueba de un requerimiento?

Para estimar el esfuerzo en crear y ejecutar casos de prueba de un requerimiento, me basaría en:

1. Complejidad y alcance del requerimiento: funcionalidades impactadas, flujos a ejecutar y cantidad de validaciones.
2. Dependencia con otros sistemas o módulos: si el requerimiento interactúa con servicios externos, bases de datos u otros componentes.
3. Cantidad y tipos de pruebas a ejecutar: determinar si se incluiran pruebas funcionales, de regresion, de integración, manuales y automatizadas.
4. Esfuerzo de diseño: tiempo destinado a diseñar los casos de prueba, preparar datos y desarrollar scripts (en caso de pruebas automatizadas)

5. Ejecución y validación de resultados.

En resumen, consideraría complejidad técnica, alcance funcional y tipo de pruebas. Es importante considerar, además, la carga de trabajo y pruebas pendientes que se tengan al momento de la estimación de un nuevo requerimiento.

Ejercicio 7:

¿Cómo harías o qué enfoque utilizarías para realizar pruebas de microservicios?

En primer lugar, ejecutaría *pruebas unitarias* para validar el funcionamiento de cada microservicio de forma aislada, asegurando que cumpla correctamente con su lógica interna y sus responsabilidades individuales.

Luego, una vez finalizadas las pruebas unitarias de forma exitosa, realizaría *pruebas de integración*, con el objetivo de verificar que los microservicios interactúan correctamente entre sí y con otros sistemas externos, garantizando la correcta comunicación mediante sus APIs o colas de mensajes.

A continuación, llevaría a cabo *pruebas de contrato*, destinadas a confirmar que cada microservicio cumple con el contrato establecido (formato, estructura y contenido de las solicitudes y respuestas), y que otros servicios pueden consumirlos sin errores ni incompatibilidades.

Finalmente, realizaría pruebas *End-to-End (E2E)* para validar flujos funcionales completos que involucren varios microservicios, asegurando que el sistema funcione de manera coherente y correcta desde la perspectiva del usuario final.

Ejercicio 8:
Siguiendo la solicitud y respuesta mostrada se solicita que genere casos de pruebas bajo la estructura: Título; Prioridad Precondiciones; Pasos; Resultados esperados, tratando de abarcar la mayoría de casos de prueba posible y explique qué es lo que busca validar en cada caso.
Esta solicitud (GET) {{LOCATION}}/bonds/{{AL30}} obtiene la siguiente respuesta:

Título	Prioridad	Precondiciones	Pasos	Resultado esperado	Objetivo
Validar respuesta exitosa con bond existente	Alta	Tener ID de un bono existente (Ej: AL30)	1. Ejecutar GET {{LOCATION}}/bonds/AL30	Respuesta 200 OK	Verificar que el endpoint devuelve la información de un bono existente.
Validar respuesta con bond inexistente	Alta	No existe un bono con el ID ingresado	1. Ejecutar GET {{LOCATION}}/bonds/ALaa	Respuesta: 404 Not Found	Verificar el manejo de errores: recurso inexistente
Validar respuesta con bond vacío	Alta	-	1. Ejecutar GET {{LOCATION}}/bonds	Respuesta: 400 Bad Request	Verificar manejo de errores: parámetro faltante
Validar respuesta con bond inválido	Media	-	1. Ejecutar GET {{LOCATION}}/bonds/!!!!	Respuesta: 400 Bad Request	Verificar manejo de errores: parámetro faltante
Validar estructura y tipo de datos en JSNO	Alta	Conocer estructura y tipo de datos esperados	1. Ejecutar GET {{LOCATION}}/bonds/AL30 2. Analizar cuerpo de la respuesta: a. shortName: string b. name: string c. debtType: string d. links: object	Los campos tienen el tipo de dato esperado	Validar formateo de los campos devueltos.
Validar campos obligatorios	Alta	Conocer campos obligatorios esperados	1. Ejecutar GET {{LOCATION}}/bonds/AL30 2. Validar que están los campos obligatorios (shortName, name, debtType, links)	Todos los campos obligatorios están presentes	Validar que la respuesta contiene los datos mínimos requeridos
Validar que shortName coincide con URL	Media	-	1. Ejecutar GET {{LOCATION}}/bonds/AL30 2. Validar que shortName coincide con el parámetro enviado en la URL.	El campo coincide con el parámetro	Verificar coherencia entre id solicitado e id devuelto
Validar tiempos de respuesta	Baja	-	1. Ejecutar GET {{LOCATION}}/bonds/AL30 varias veces 2. Medir el tiempo de respuesta 3. Considerar tiempo promedio	El tiempo promedio es menor a 2 segundos	Validar rapidez en respuesta del endpoint.

Ejercicio 9:

Se solicita diseñar casos automatizados utilizando cypress y la página https://www.saucedemo.com/v1/index.html. Explique el paso a paso de como fue pensando y generando el test. Se deberá subir el proyecto a un repositorio público de github para poder realizar la corrección desde ahí. Considere utilizar las buenas prácticas de cypress para resolverlo.

Se entrega en el repositorio una carpeta “Sauce Demo - Cypress Tests” correspondiente a la resolución de esta consigna.

Ejercicio 10:

Se solicita diseñar casosArmar un smoke test sobre una API pública utilizando Postman. Las pruebas deben enfocarse en comprobar que los endpoints principales responden correctamente y devuelven los datos esperados.

Se entregan dentro del repositorio dos archivos:

- *Belardinelli - SimpsonsAPI Smoke Test.postman_collection.json*
- *Belardinelli - SimpsonsAPI.postman_environment.json*

Correspondientes a la resolución de esta consigna.