Informe de Ejecución del Plan de Pruebas

Proyecto de Pruebas: Plan de Prueba Numero 3
Plan de Pruebas: Plan de Testing Prueba QA 2

Imprimido por TestLink el 20/10/2023

Proyecto de Pruebas: Plan de Prueba Numero 3

Plan de pruebas manuales para API REST con SpringBoot, diseñada por la Desarrolladora Full Stack Nicole Opazo.

Se verifican todas las respuestas con los diferentes métodos (GET-POST-PUT-DELETE) en POSTMAN.

Se verifica la respuesta al envío de los Endpoints señalados en README del repositorio.

Tester: Antonella Muñoz C.

Plan de Pruebas: Plan de Testing Prueba QA 2

PLAN DE TEST DEL SISTEMA

API REST desarrollada con SpringBoot.

PREFACIO

Este documento describe el plan de test del sistema del proyecto API REST, cuyo objetivo principal es verificar su correcto funcionamiento a través de pruebas manuales.

Alcance Este documento de plan de test es la base de la fase de test del proyecto.

HISTORIA DEL DOCUMENTO

Fecha Versión Comentarios Autor

5 de Octubre

2023 1.0 Versión inicial

1.1 Revisada por el equipo

TABLA DE CONTENIDOS

1 INTRODUCCION

1.1 Propósito

Se desarrolló un proyecto API REST en Java, la cual gestiona dos entidades: Usuario y Carrito. Utiliza una base de datos H2 en memoria y ofrece varios endpoints para realizar operaciones CRUD en estas entidades.

1.2 Objetivos del Plan

EL Plan de Test del Sistema especifica los procesos de test y de verificación que se realizaran con el objeto de:

· Identificar defectos y fallas.

· Medir rendimiento.

· Evaluar la calidad

· Determinar el cumplimiento de los requerimientos.

Los objetivos de este plan son:

· Definir y detallar toadas las tareas que se desarrollarán para probar el sistema.

· Definir el plan y la persona o grupo responsable de cada tarea.

· Definir las herramientas de prueba y el ambiente necesario a la conducción de las actividades de test.

· Definir los ítems y funcionalidades que serán probados.

1.3 Alcance del Testing

El Plan de Testing del Sistema es una especificación de alto nivel de los requerimientos funcionales y de calidad que serán probados, del ambiente de testing, de la estrategia de testing, de las responsabilidades y de los criterios de éxito.

El comportamiento de un producto bajo testing sera comparado con las especificaciones de los requerimientos que fueron usados para implementar el sistema, incluyendo todos los cambios que han sido aprobados e implementados.

Los casos de prueba y los criterios de éxito serán derivados de este plan general y serán especificados en el documento de Especificaciones de Testing del Sistema.

El alcance del test del sistema es probar la funcionalidad completa y el rendimiento del proyecto API-REST

1.4 Criterios de Entrada

Para poder comenzar la fase de pruebas del sistema, se deben cumplir los siguientes criterios:

· Test unitarios realizados y completados para cada componente del sistema.

· Sistema completamente integrado.

· Software congelado.

· Hardware congelado.

1.5 Criterios de Salida

1. **Todos los Casos de Prueba Ejecutados:** Todos los casos de prueba planificados han sido ejecutados con éxito sin errores graves y se han documentado los resultados de cada prueba.
2. **Requisitos Funcionales Cumplidos:** Se ha verificado que todas las funciones y características especificadas en los requisitos funcionales del proyecto se han implementado y funcionan según lo esperado.
3. **Requisitos No Funcionales Cumplidos:** Se han evaluado y verificado los requisitos no funcionales, como el rendimiento, la escalabilidad y la seguridad, y se han cumplido según los criterios definidos.
4. **Rendimiento Aceptable:** El sistema ha sido sometido a pruebas de rendimiento y carga, y ha demostrado que puede manejar la carga prevista sin degradación significativa del rendimiento.
5. **Documentación Actualizada:** La documentación del sistema, incluidos manuales de usuario y cualquier documentación técnica, se ha actualizado para reflejar los cambios realizados durante el proceso de desarrollo y pruebas.
6. **Errores Corregidos:** Todos los errores críticos identificados durante las pruebas han sido corregidos, verificados y cerrados.
7. **Aprobación del Cliente:** El cliente o los interesados han revisado y aprobado la funcionalidad del sistema y están satisfechos con los resultados de las pruebas.
8. **Informe de Pruebas:** Se ha generado un informe de pruebas completo que documenta los resultados de todas las pruebas realizadas, incluyendo casos de prueba, resultados, problemas encontrados y acciones tomadas.
9. **Entorno de Pruebas Restaurado:** Cualquier configuración o entorno de pruebas utilizado se ha restaurado a su estado original y se encuentra disponible para futuras pruebas o referencia.
10. **Plan de Pruebas Actualizado:** El plan de pruebas se ha actualizado para incluir cualquier cambio o desviación que haya surgido durante la fase de pruebas.

Una vez que se cumplan estos criterios de salida, se considerará que la fase de pruebas del sistema ha sido exitosa y se podrá proceder con la siguiente etapa del proyecto.

2 Test del Sistema

El sistema a probar se refiere a la API REST desarrollada en Spring Boot denominada "ApiRestSpringBoot". Esta API está diseñada para gestionar dos entidades principales: Usuario y Carrito. La API utiliza una base de datos H2 en memoria y ofrece una serie de endpoints que permiten realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en estas entidades.

Límites del Sistema:

1. **Entidades de Usuario y Carrito:** El sistema se centra en la gestión de dos entidades principales: Usuario y Carrito. Estas entidades representan usuarios y sus carritos de compras respectivamente.
2. **Operaciones CRUD:** El sistema proporciona endpoints para realizar operaciones CRUD en estas entidades, lo que incluye la creación, lectura, actualización y eliminación de registros.
3. **Base de Datos H2 en Memoria:** El sistema utiliza una base de datos H2 en memoria para almacenar los datos de Usuario y Carrito. Esto significa que los datos se mantienen solo durante la ejecución de la aplicación y no se almacenan de manera persistente en un sistema de base de datos externo.
4. **API RESTful:** La API sigue los principios de una arquitectura RESTful, lo que significa que utiliza los métodos HTTP (GET, POST, PUT, DELETE) y URLs para interactuar con los recursos.
5. **Acceso a través de HTTP:** Los clientes pueden interactuar con el sistema a través de solicitudes HTTP a las URL proporcionadas por la API.
6. **Rutas de Endpoints:** El sistema define rutas de endpoints para cada operación, como la creación de un usuario o la obtención de un carrito por su ID.

7. **Interfaz de Usuario:** La API no tiene una interfaz de usuario propia, ya que se espera que los clientes interactúen con ella a través de solicitudes HTTP.

8. **Documentación:** La API está documentada en este archivo y se proporcionan ejemplos de solicitudes y respuestas para cada uno de sus endpoints.

El objetivo del test del sistema es garantizar que la API "ApiRestSpringBoot" funcione de acuerdo con los requisitos especificados, que pueda gestionar los datos de Usuario y Carrito de manera adecuada y que responda correctamente a las solicitudes HTTP realizadas por los clientes.

Se probarán casos de prueba que abarcan las operaciones CRUD, la gestión de errores, la seguridad y el rendimiento del sistema. Además, se verificará que la API cumpla con los requisitos funcionales y no funcionales establecidos para el proyecto.

2.1 Estrategia de Test del Sistema

La estrategia de prueba del sistema para la API REST "ApiRestSpringBoot" se basará en la ejecución de una serie de casos de prueba que abarcarán diferentes aspectos de la funcionalidad y el rendimiento del sistema. A continuación, se describe la estrategia general de prueba:

Entradas:

1. **Solicitudes HTTP:** Se enviarán solicitudes HTTP a los diferentes endpoints de la API utilizando varios métodos HTTP (GET, POST, PUT, DELETE). Estas solicitudes simularán las acciones de los clientes.

2. **Datos de Prueba:** Se proporcionarán datos de prueba en formato JSON para crear, actualizar y eliminar registros de Usuario y Carrito. Estos datos reflejarán diferentes escenarios y configuraciones posibles.

3. **Criterios de Prueba:** Se definirán criterios de prueba específicos para cada caso de prueba, que establecerán las condiciones y resultados esperados.

Salidas Observadas:

1. **Respuestas HTTP:** Se observarán las respuestas HTTP generadas por la API en respuesta a las solicitudes. Se verificará que las respuestas sean coherentes con los criterios de prueba y que incluyan los datos correctos.

2. **Estado de la Base de Datos:** Se verificará el estado de la base de datos H2 en memoria para asegurarse de que las operaciones de creación, actualización y eliminación se reflejen correctamente en los registros de Usuario y Carrito.

3. **Rendimiento:** Se medirá el rendimiento del sistema al evaluar su capacidad para manejar múltiples solicitudes concurrentes y su tiempo de respuesta en diferentes situaciones.

4. **Errores y Excepciones:** Se observarán cualquier error o excepción generados por la API y se comprobará que se gestionen adecuadamente, proporcionando mensajes de error comprensibles.

Estrategia de Prueba:

La estrategia de prueba se centrará en los siguientes aspectos clave:

1. **Pruebas de Operaciones CRUD:** Se probarán todas las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en las entidades de Usuario y Carrito. Esto incluye la creación de nuevos registros, la lectura de registros existentes, la actualización de registros y la eliminación de registros.

2. **Pruebas de Rutas de Endpoints:** Se verificará que las rutas de los endpoints estén correctamente configuradas y que las solicitudes a estas rutas sean manejadas de acuerdo con la documentación proporcionada.

3. **Pruebas de Seguridad:** Se comprobará la seguridad de la API mediante la validación de datos de entrada, la autenticación y autorización de usuarios, y la protección contra posibles ataques, como la inyección SQL.

4. **Pruebas de Rendimiento:** Se realizarán pruebas de carga para evaluar el rendimiento de la API bajo condiciones de uso intensivo, incluyendo pruebas de alta concurrencia.

5. **Pruebas de Errores y Excepciones:** Se verificará que la API maneje adecuadamente los errores y excepciones, proporcionando respuestas de error claras y adecuadas.

6. **Pruebas de Escenarios Específicos:** Se probarán escenarios específicos que reflejen situaciones de uso real, como la obtención de un carrito de usuario o la actualización de un usuario existente.

La estrategia de prueba se llevará a cabo de manera manual, siguiendo los casos de prueba definidos para cada uno de los aspectos mencionados anteriormente. Se documentarán los resultados de las pruebas, incluyendo cualquier defecto encontrado y su gravedad. Además, se proporcionará evidencia de la ejecución de las pruebas y se registrarán los tiempos de respuesta observados durante las pruebas de rendimiento.

2.2 Pruebas en Operación Normal

Durante la operación normal del sistema API REST "ApiRestSpringBoot", se llevarán a cabo las siguientes categorías de pruebas para evaluar su funcionalidad:

1. **Pruebas de Creación de Usuario y Carrito:**
    - Verificar la capacidad del sistema para crear nuevos registros de Usuario y Carrito.
    - Comprobar que los datos proporcionados en las solicitudes de creación se almacenan correctamente en la base de datos.
2. **Pruebas de Lectura de Usuario y Carrito:**
    - Validar la capacidad del sistema para recuperar registros de Usuario y Carrito existentes.
    - Comprobar que las solicitudes de lectura devuelvan los datos correctos de acuerdo con los identificadores proporcionados.
3. **Pruebas de Actualización de Usuario y Carrito:**
    - Evaluar la capacidad del sistema para actualizar información en registros de Usuario y Carrito existentes.
    - Comprobar que las solicitudes de actualización reflejen correctamente los cambios en la base de datos.
4. **Pruebas de Eliminación de Usuario y Carrito:**
    - Verificar que el sistema pueda eliminar registros de Usuario y Carrito existentes.
    - Comprobar que las solicitudes de eliminación eliminen los registros correspondientes de la base de datos.
5. **Pruebas de Rutas de Endpoints:**
    - Validar que todas las rutas de los endpoints estén configuradas correctamente y sean accesibles.
    - Comprobar que las solicitudes a las rutas de los endpoints se gestionen adecuadamente.
6. **Pruebas de Seguridad:**
    - Evaluar la seguridad del sistema mediante la validación de datos de entrada y la prevención de ataques.
    - Comprobar que la autenticación y autorización de usuarios se implementen correctamente.
7. **Pruebas de Rendimiento:**
    - Medir el rendimiento del sistema bajo condiciones de uso normal.
    - Evaluar el tiempo de respuesta de las solicitudes y la capacidad del sistema para manejar cargas de trabajo típicas.
8. **Pruebas de Errores y Excepciones:**
    - Verificar que el sistema maneje adecuadamente los errores y excepciones, proporcionando mensajes de error claros y coherentes.
9. **Pruebas de Escenarios Específicos:**
    - Probar escenarios específicos que reflejen situaciones de uso real, como la obtención de un carrito de usuario o la actualización de un usuario existente.

Estas categorías de pruebas cubrirán aspectos esenciales del funcionamiento de la API y garantizarán que opere de manera confiable y eficiente durante su operación normal.

2.2.1 Funcionalidad Grupo 1

Este grupo se enfocará en evaluar las funcionalidades relacionadas con la creación y lectura de registros de Usuario y Carrito durante la operación normal del sistema.

Funcionalidad: Creación de Usuario y Carrito

- El sistema debe permitir la creación de nuevos registros de Usuario y Carrito.
- Esta funcionalidad se considera exitosa si, al enviar una solicitud POST con datos válidos, se crea un nuevo usuario o carrito en la base de datos y se devuelve una respuesta HTTP 201 (Created).
- Se considera un fracaso si no se crea el registro o si se devuelve un código de estado HTTP incorrecto.

- **Funcionalidad: Lectura de Usuario y Carrito**

- El sistema debe permitir la lectura de registros de Usuario y Carrito por su ID.
- Esta funcionalidad se considera exitosa si, al enviar una solicitud GET con un ID válido, se obtiene la información correcta del usuario o carrito y se devuelve una respuesta HTTP 200 (OK).
- Se considera un fracaso si no se obtiene la información correcta o si se devuelve un código de estado HTTP incorrecto.

2.3 Pruebas en Condiciones de Excepción

El sistema debe ser capaz de manejar diversas condiciones de excepción. Aquí se mencionan algunas condiciones de excepción que se probarán:

2.3.1 Condición de Excepción 1: **Solicitud de Usuario Inexistente**

- Descripción: Se enviará una solicitud GET para obtener un usuario que no existe en la base de datos.
- Criterio de Éxito: El sistema debe devolver una respuesta HTTP 404 (Not Found).
- Criterio de Fracaso: El sistema devuelve un código de estado HTTP incorrecto.

2.3.2 Condición de Excepción: Solicitud de Actualización con Datos Incorrectos

- Descripción: Se enviará una solicitud PUT para actualizar un usuario con datos incorrectos o incompletos.
- Criterio de Éxito: El sistema debe validar los datos y rechazar la solicitud, devolviendo un código de estado HTTP 400 (Bad Request).
- Criterio de Fracaso: El sistema acepta la solicitud y actualiza el usuario con datos incorrectos.

2.4 Criterios de Éxito/Fracaso de Pruebas

- Una prueba individual se considera exitosa si cumple con los criterios establecidos para esa prueba específica (por ejemplo, la creación de un usuario devuelve un código de estado HTTP 201 y crea el registro correctamente).
- Un grupo de pruebas se considera exitoso si todas las pruebas individuales dentro del grupo se completan con éxito.
- Se asigna el valor "Completamente Pasado" cuando todas las pruebas individuales en un grupo se pasan con éxito.
- Se asigna el valor "Parcialmente Pasado" si algunas pruebas individuales en un grupo fallan pero otras pasan con éxito.
- Se asigna el valor "No Pasado" si todas las pruebas individuales en un grupo fallan.
- Estos criterios de éxito y fracaso se aplicarán a cada prueba individual y a cada grupo de pruebas.

2.5 Entregables

Los resultados de las pruebas se documentarán en un informe de prueba que contendrá los detalles de las pruebas realizadas, los resultados obtenidos y cualquier observación relevante. El informe estará disponible en formato electrónico y se compartirá con el equipo de desarrollo y otros interesados relevantes.

3 Configuración del Test

Esta sección establece los componentes del ambiente de testing

3.1 Hardware

- Computadora o servidor compatible con Spring Boot y Java.
- Conexión a Internet para acceder a recursos externos si es necesario.

3.2 Software

- Java JDK instalado.
- Apache Maven para compilar y ejecutar el proyecto.
- IDE o editor de código (por ejemplo, Visual Studio Code o IntelliJ IDEA) para trabajar con el código fuente si es necesario.
- Git para clonar el repositorio.
- Postman u otra herramienta de prueba de API para enviar solicitudes HTTP y verificar las respuestas.
- Windows 10

3.3 Otros

Datos de prueba, como JSON de solicitud de ejemplo para crear usuarios y carritos.

3.4 Ambiente

El ambiente de prueba consiste en:

- Un servidor local o en la nube donde se ejecute la aplicación Spring Boot.
- Una máquina de desarrollo o prueba con las herramientas mencionadas instaladas.
- Conexión a Internet para acceder a recursos externos si es necesario.

4 Tareas

4.1 Actividades

La secuencia de actividades para probar el sistema es:

1. **Preparación del Ambiente de Prueba**:
    - Obtener el hardware necesario.

- Instalar y configurar el software necesario.
- Clonar el repositorio del proyecto desde Git.

2. **Desarrollo de Pruebas Manuales**:
    - Identificar y diseñar casos de prueba para las funcionalidades de la API.
    - Crear datos de prueba, incluyendo JSON de solicitud de ejemplo.
    - Realizar pruebas manuales siguiendo los casos de prueba.
3. **Registro de Resultados**:
    - Registrar los resultados de las pruebas, incluyendo éxitos y fracasos.
    - Documentar cualquier problema o observación relevante.
4. **Reporte de Defectos**:
    - En caso de encontrar defectos, documentarlos en un sistema de seguimiento de problemas si está disponible.
5. **Informe de Prueba**:
    - Generar un informe de prueba que resuma los resultados de las pruebas y proporcione una evaluación general de la calidad del sistema probado.
6. **Revisión y Validación**:
    - Revisar el informe de prueba y validar que todas las pruebas se realizaron correctamente.
7. **Entrega de Resultados**:
    - Compartir el informe de prueba con el equipo de desarrollo y otros interesados relevantes.
8. **Cierre de Pruebas**:
    - Realizar cualquier limpieza o cierre necesario en el ambiente de prueba.
9. **Retrospectiva de Pruebas**:
    - Realizar una retrospectiva para identificar áreas de mejora en el proceso de pruebas.

Estas actividades se llevarán a cabo de manera secuencial para garantizar que el sistema se pruebe exhaustivamente y que los resultados de las pruebas se documenten adecuadamente.

4.2 Responsabilidades

Esta sección establece las responsabilidades de cada grupo queparticipa en la fase de pruebas.

4.2.1 Responsabilidades del Grupo de Desarrollo

· Ejecutar las pruebas unitarias

· Ejecutar y probar la integración de bajo nivel

· Corregir los problemas reportados

4.2.2 Responsabilidades del Grupo deTesting

· Planificar las pruebas del sistema

· Configurar el ambiente de prueba

· Ejecutar las pruebas del sistema

· Escribir el reporte de test

4.2.3 Responsabilidades de la Gerencia

· Proveer recursos

· Aceptación final y aprobación de la liberación del producto

4.3 Planificación

La planificación estimada de las actividades de testing se llevará a cabo durante las próximas dos semanas, siguiendo la secuencia de actividades descritas en la sección 4.1. Esto permitirá realizar pruebas exhaustivas en el sistema API REST Spring

Boot y garantizar su correcto funcionamiento.

Glosario (Definiciones y Siglas)

- API: Interfaz de Programación de Aplicaciones.
- CRUD: Create, Read, Update, Delete (Crear, Leer, Actualizar, Eliminar).
- JSON: Notación de Objetos JavaScript (JavaScript Object Notation).
- Git: Sistema de control de versiones distribuido.
- IDE: Entorno de Desarrollo Integrado (Integrated Development Environment).
- Maven: Herramienta de gestión de proyectos de software.
- H2: Base de datos en memoria.
- URL: Localizador Uniforme de Recursos (Uniform Resource Locator).
- HTTP: Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol).

Suite de Pruebas : Crear un usuario

Caso de Prueba pqa3-1: Registro exitoso de usuario [Versión : 1]			
Nº:	Pasos:	Resultados Esperados:	Estado de la ejecución:
1	Modificar URL añadiendo : /usuarios	http://localhost:8080/usuarios	Pasado
2	Seleccionar en el menú que está bajo la URL la opción "Body". Hacer clic en la opción "raw" y luego aparecerá un menú con la palabra "TEXT". Cambiar "TEXT" por "JSON		Pasado
3	Pegar en "Body" el siguiente Endpoint, modificando sólo los datos: { "nombre": "usuario", "edad": 26, "activo": true, "fechaCreacion": "2023-09-28T10:00:00", "carrito": { "codigo": "CART001", "total": 100.00, "fechaCompra": "2023-09-28T10:00:00" } }	{ "nombre": "antmunozc", "edad": 29, "activo": true, "fechaCreacion": "2023-10-20T10:00:00", "carrito": { "codigo": "CART001", "total": 300.00, "fechaCompra": "2023-09-28T10:00:00" } }	Pasado
4	Hacer clic en botón SEND	Código "200 OK" Usuario creado de forma exitosa.	Pasado
Tipo de ejecución:	Manual		
Duración estimada de la ejec. (min):	2.00		
Prioridad:	Alta		
Detalles de la ejecución			
Build	V 2.0		
Tester	user		
Resultado de la Ejecución:	**Pasado**		
Modo de Ejecución:	**Manual**		
Duración de le ejecución (min):			
Notas de la Ejecución	Test realizado de forma exitosa.		

Caso de Prueba pqa3-2: Registro de usuario con edad no válida FAIL [Versión : 1]			
Nº:	Pasos:	Resultados Esperados:	Estado de la ejecución:
1	Modificar URL añadiendo : /usuarios	http://localhost:8080/usuarios	Pasado
2	Seleccionar en el menú que está bajo la URL la opción "Body". Hacer clic en la opción "raw" y luego aparecerá un menú con la palabra "TEXT". Cambiar "TEXT" por "JSON		Pasado
3	Pegar en "Body" el siguiente Endpoint, modificando sólo los datos. En "edad"	{ "nombre": "antmunozc", "edad": "hola", "activo": true, "fechaCreacion": "2023-09-28T10:00:00",	Pasado

	agregar respuesta alfabética: { "nombre": "usuario", "edad": 26, "activo": true, "fechaCreacion": "2023-09-28T10:00:00", "carrito": { "codigo": "CART001", "total": 100.00, "fechaCompra": "2023-09-28T10:00:00" } }	"carrito": { "codigo": "CART001", "total": 100.00, "fechaCompra": "2023-09-28T10:00:00" } }	
4	Hacer clic en botón SEND.	ERROR 400. BAD REQUEST. Intento fallido. No permite crear el usuario.	Pasado

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	
Prioridad:	Media

Detalles de la ejecución	
Build	V 2.0
Tester	user
Resultado de la Ejecución:	**Fallado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	El fallo de este test nos muestra un aspecto positivo en el desarrollo de la aplicación, puesto que impide que un usuario registre datos no veraces o que por error no señale bien la información solicitada.

Suite de Pruebas : Eliminar un usuario

Caso de Prueba pqa3-5: Eliminar de forma exitosa un usuario [Versión : 1]			
Nº:	Pasos:	Resultados Esperados:	Estado de la ejecución:
1	Añadir en campo URL : /usuarios/{id}	Debe quedar así http://localhost:8080/usuarios/{id}	Pasado
2	Cambiar "id" por un número específico de usuario.	Debe quedar así http://localhost:8080/usuarios/1	Pasado
3	Hacer clic en botón SEND.	Mensaje en respuesta que usuario fue eliminado.	Pasado
Tipo de ejecución:	Manual		
Duración estimada de la ejec. (min):			
Prioridad:	Media		
Detalles de la ejecución			
Build	V 2.0		
Tester	user		
Resultado de la Ejecución:	**Pasado**		
Modo de Ejecución:	**Manual**		
Duración de le ejecución (min):			
Notas de la Ejecución	Test aplicado de forma exitosa		

Caso de Prueba pqa3-6: Corroborar que usuario fue eliminado [Versión : 1]			
Nº:	Pasos:	Resultados Esperados:	Estado de la ejecución:
1	Modificar URL añadiendo : /usuarios/{id}	Debería quedar de esta forma http://localhost:8080/usuarios/{id}	Pasado
2	Modificar "id" por número id de usuario que fue eliminado	Si el id de usuario eliminado era "1", debería quedar de esta forma: http://localhost:8080/usuarios/1	Pasado
3	Haga clic en botón SEND.	Debería aparecer en la respuesta que usuario no existe.	Pasado
Tipo de ejecución:	Manual		
Duración estimada de la ejec. (min):			
Prioridad:	Media		
Detalles de la ejecución			
Build	V 2.0		
Tester	user		
Resultado de la Ejecución:	**Pasado**		
Modo de Ejecución:	**Manual**		

Duración de le ejecución (min):	
Notas de la Ejecución	Test aplicado de forma exitosa

Suite de Pruebas : Obtener Información de Usuario

Caso de Prueba pqa3-7: Obtener información de un usuario [Versión : 1]			
Nº:	Pasos:	Resultados Esperados:	Estado de la ejecución:
1	Modificar la URL añadiendo "/usuarios/{id}"	Debería verse de esta forma: http://localhost:8080/usuarios/{id}	Pasado
2	Modificar "id" por número de ID concreto de usuario a consultar. Si el ID es 1, se reemplaza por ese número.	Debería verse de esta forma: http://localhost:8080/usuarios/1	Pasado
3	Hacer Clic en botón SEND.	Código 200. Se muestra en la respuesta los datos del usuario consultado.	Pasado
Tipo de ejecución:	Manual		
Duración estimada de la ejec. (min):			
Prioridad:	Media		
Detalles de la ejecución			
Build	V 2.0		
Tester	user		
Resultado de la Ejecución:	**Pasado**		
Modo de Ejecución:	**Manual**		
Duración de le ejecución (min):			
Notas de la Ejecución	Test realizado de forma exitosa		

Caso de Prueba pqa3-8: Obtener información de todos los usuarios [Versión : 1]			
Nº:	Pasos:	Resultados Esperados:	Estado de la ejecución:
1	Modificar URL añadiendo "/usuarios/todos"	Debe quedar de esta forma: http://localhost:8080/usuarios/todos	Pasado
2	Hacer clic en botón SEND.	Respuesta cód 200. Muestra todos los usuarios registrados.	Pasado
Tipo de ejecución:	Manual		
Duración estimada de la ejec. (min):			
Prioridad:	Media		
Detalles de la ejecución			
Build	V 2.0		
Tester	user		
Resultado de la Ejecución:	**Pasado**		
Modo de Ejecución:	**Manual**		
Duración de le ejecución (min):			
Notas de la Ejecución	Test realizado de forma exitosa.		

Suite de Pruebas : Actualizar datos

Caso de Prueba pqa3-9: Actualizar un Usuario [Versión : 1]			
Nº:	Pasos:	Resultados Esperados:	Estado de la ejecución:
1	Modificar la URL añadiendo : /usuarios/{id}	http://localhost:8080/usuario/{id}	Pasado
2	Modificar campo {id} por número de ID de usuario específico. Ejemplo: Si el ID es 1, se modifica por ese número.	http://localhost:8080/usuario/1	Pasado
3	Seleccionar bajo la URL, la pestaña "Body" y cambiar el menú "Text" a "JSON"		Pasado
4	Utilizar en body el siguiente Endpoint, modificando sólo los datos: { "nombre": "usuario_actualizado", "edad": 27, "activo": true, "fechaCreacion": "2023-09-27T11:00:00" }	{ "nombre": "antmunozc", "edad": 29, "activo": true, "fechaCreacion": "2023-10-20T11:00:00" }	Pasado
5	Hacer clic en botón SEND	Código 200 OK. Datos actualizados	Pasado
Tipo de ejecución:	Manual		
Duración estimada de la ejec. (min):			
Prioridad:	Media		
Detalles de la ejecución			
Build	V 2.0		
Tester	user		
Resultado de la Ejecución:	**Pasado**		
Modo de Ejecución:	**Manual**		
Duración de le ejecución (min):			
Notas de la Ejecución	Test realizado de forma exitosa.		

Caso de Prueba pqa3-10: Actualizar un carrito [Versión : 1]			
Nº:	Pasos:	Resultados Esperados:	Estado de la ejecución:
1	Modificar la URL añadiendo : /carritos/{idCarrito}	http://localhost:8080/carritos/{idCarrito}	Pasado
2	Modificar campo {id} por número de ID de carrito específico. Ejemplo: Si el ID es 1, se modifica por ese	http://localhost:8080/carritos/1	Pasado
3	Seleccionar bajo la URL, la pestaña "Body" y cambiar el menú "Text" a "JSON"		Pasado

4	Utilizar en body el siguiente Endpoint, modificando sólo los datos: { "codigo": "CART001_actualizado", "total": 200.00, "fechaCompra": "2023-09-28T11:00:00" }	{ "codigo": "CART005_actualizado", "total": 300, "fechaCompra": "2023-03-28T11:00:00" }	Pasado
5	Hacer clic en botón SEND.	Código 200 OK. Datos actualizados.	Pasado

Tipo de ejecución:	Manual
Duración estimada de la ejec. (min):	
Prioridad:	Media
Detalles de la ejecución	
Build	V 2.0
Tester	user
Resultado de la Ejecución:	**Pasado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	Test realizado de forma exitosa.

Suite de Pruebas : Obtener Información de carrito

Caso de Prueba pqa3-11: Obtener un carrito por su ID [Versión : 1]			
Nº:	Pasos:	Resultados Esperados:	Estado de la ejecución:
1	Modificar URL añadiendo: /carritos/{idCarrito}	http://localhost:8080/carritos/{idCarrito}	Pasado
2	Modificar {idCarrito} por número de ID específico a consultar. Si es 1, se reemplaza por ese número.	http://localhost:8080/carritos/1	Pasado
3	Clic en botón SEND.	Respuesta código 200 OK. Deberían mostrarse los datos del carro.	Pasado
Tipo de ejecución:	Manual		
Duración estimada de la ejec. (min):			
Prioridad:	Media		
Detalles de la ejecución			
Build	V 2.0		
Tester	user		
Resultado de la Ejecución:	**Pasado**		
Modo de Ejecución:	**Manual**		
Duración de le ejecución (min):			
Notas de la Ejecución	Test realizado de forma exitosa.		

Caso de Prueba pqa3-12: Obtener el carrito de un usuario [Versión : 1]			
Nº:	Pasos:	Resultados Esperados:	Estado de la ejecución:
1	Modificar URL añadiendo: /carritos/usuario/{idUsuario}	http://localhost:8080/carritos/usuario/{idUsuario}	Pasado
2	Modificar {idUsuario} por número especifico de ID de Usuario a consultar. Ej: Si ID de usuario es 1, se reemplaza por ese número.	http://localhost:8080/carritos/usuario/1	Pasado
3	Clic en botón SEND	Código 200 OK. Respuesta muestra datos de carrito asociado a usuario.	Pasado
Tipo de ejecución:	Manual		
Duración estimada de la ejec. (min):			
Prioridad:	Media		
Detalles de la ejecución			
Build	V 2.0		
Tester	user		

Resultado de la Ejecución:	**Pasado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	Test realizado de forma exitosa.

Caso de Prueba pqa3-13: Obtener todos los carritos [Versión : 1]			
Nº:	Pasos:	Resultados Esperados:	Estado de la ejecución:
1	Modificar URL añadiendo /carritos/todoscarritos	http://localhost:8080/carritos/todoscarritos	Pasado
2	Clic en botón SEND	Cód. 200 OK. Respuesta debería mostrar los datos de todos los carritos existentes.	Pasado
Tipo de ejecución:	Manual		
Duración estimada de la ejec. (min):			
Prioridad:	Media		

Detalles de la ejecución	
Build	V 2.0
Tester	user
Resultado de la Ejecución:	**Pasado**
Modo de Ejecución:	**Manual**
Duración de le ejecución (min):	
Notas de la Ejecución	Test realizado de forma exitosa.