Calidad de Software y Plan de Pruebas

Desarrollo de Software

Tabla de contenido

Objetivo

El objetivo principal de este plan de pruebas es diseñar y ejecutar pruebas de funcionalidad que aseguren que todos los componentes críticos del sistema operan conforme a los requisitos establecidos. Esto abarca verificar que los usuarios acceder a la aplicación, con todas las medidas de seguridad pertinentes, y que el sistema gestione correctamente los datos de prácticas profesionales.

Calidad del Software

Para garantizar la entrega de una API de alta calidad, hemos implementado un riguroso proceso de aseguramiento y control de calidad. Se han llevado a cabo pruebas exhaustivas, incluyendo pruebas unitarias, de integración y funcionales, utilizando herramientas como JUnit y Selenium. Además, se han aplicado estándares de codificación y se han realizado revisiones de código de manera regular.

A través de la automatización de pruebas, hemos logrado reducir significativamente el tiempo de ejecución de las pruebas y mejorar la cobertura de código. Asimismo, se ha implementado un sistema de seguimiento de defectos para garantizar que todos los problemas sean identificados y resueltos de manera oportuna.

Como parte de nuestro compromiso con la mejora continua, realizamos análisis periódicos de las métricas de calidad para identificar áreas de mejora y ajustar nuestras prácticas de desarrollo en consecuencia.

Alcance:

El alcance de un plan de pruebas para una API define los límites y el enfoque de las pruebas que se realizarán. Es esencial establecer un alcance claro para garantizar que se cubran todos los aspectos críticos de la API y evitar realizar pruebas innecesarias.

El alcance del plan de pruebas de una API generalmente incluye:

Funcionalidades:

- Operaciones básicas: Crear, leer, actualizar y eliminar recursos (CRUD).
- Flujos de negocio: Simular escenarios de uso reales de la API.
- Validación de datos: Verificar que la API maneje correctamente datos válidos e inválidos.
- Autenticación y autorización: Probar diferentes mecanismos de autenticación y autorización.
- Manejo de errores: Verificar que la API devuelva mensajes de error claros y concisos.

No Funcionalidades:

- Rendimiento: Medir el tiempo de respuesta, la capacidad de carga y la escalabilidad de la API.
- Seguridad: Identificar vulnerabilidades como inyecciones SQL, XSS, etc.
- Usabilidad: Evaluar la facilidad de uso de la documentación de la API.
- Compatibilidad: Verificar que la API funcione correctamente en diferentes entornos y navegadores.

Entornos de prueba:

- Entorno de desarrollo: Para pruebas iniciales y rápidas.
- Entorno de pruebas: Para pruebas más exhaustivas y simulaciones de producción.
- Entorno de producción: Para pruebas finales y verificación de la integración con otros sistemas.

Factores que influyen en el alcance:

- Complejidad de la API: Cuanto más compleja sea la API, mayor será el alcance de las pruebas.
- Requisitos del cliente: Los requisitos específicos del cliente determinarán las áreas a las que se dará mayor prioridad.
- Restricciones de tiempo y presupuesto: El tiempo y los recursos disponibles limitarán el alcance de las pruebas.
- Riesgos asociados a la API: Se priorizarán las áreas con mayor riesgo de fallos.

Desarrollo del Plan de Pruebas

Preparación del Entorno de Pruebas:

- Configuración de un entorno de pruebas que replique las condiciones operativas del entorno de producción.
- Garantizar la disponibilidad de las herramientas necesarias para las pruebas automatizadas y manuales.

Plan de pruebas para API

Es un documento que detalla los objetivos, alcance, recursos y procedimientos que se seguirán para evaluar si una API cumple con los requisitos funcionales y no funcionales establecidos.

¿Por qué utilizar Postman?

Postman es una herramienta muy popular para el desarrollo y pruebas de APIs, ya que permite:

- Crear solicitudes HTTP: Simular cualquier tipo de petición HTTP (GET, POST, PUT, DELETE, etc.)
- Verificar respuestas: Analizar los códigos de estado, encabezados y cuerpo de las respuestas.
- Automatizar pruebas: Crear colecciones de solicitudes y ejecutarlas de forma automatizada.
- Gestionar entornos: Definir diferentes entornos de prueba con valores de variables personalizados.

Plan de pruebas para API con Postman:

Características de la API:

- Descripción general de la API
- Endpoints disponibles
- Métodos HTTP soportados
- Formato de datos (JSON, XML)
- Autenticación requerida

Escenarios de prueba:

- Casos de uso positivos (comportamiento esperado)

- Casos de uso negativos (comportamiento ante entradas inválidas)
- Pruebas de límites (valores máximos y mínimos)
- Pruebas de rendimiento (carga, estrés)
- Pruebas de seguridad (injección SQL, XSS, etc.)

Datos de prueba:

- Datos de entrada válidos e inválidos
- Valores límite
- Datos de prueba para escenarios de carga

Entorno de prueba:

- Herramientas: Postman, base de datos, servidor de aplicaciones
- Configuración del entorno
- Datos de prueba precargados

Procedimientos de prueba:

- Pasos a seguir para ejecutar cada caso de prueba
- Verificación de resultados esperados
- Captura de evidencias

Criterios de aceptación:

Condiciones que deben cumplirse para considerar que la API ha sido probada con éxito

Entorno de Prueba:

Hardware: Especificar los equipos y dispositivos en los que se realizarán las pruebas (ordenadores, dispositivos móviles).

Software: Especificar el sistema operativo, el navegador y las herramientas de prueba a utilizar.

Datos: Especificar los datos de prueba necesarios (usuarios válidos, usuarios inválidos, contraseñas).

Recursos:

Personal: Testers manuales, programadores automatizados.

Herramientas: Herramientas de gestión de pruebas, herramientas de automatización (Selenium, Cypress).

Riesgos:

- Defectos en la lógica de autenticación.
- Vulnerabilidades de seguridad.
- Problemas de rendimiento en condiciones de alta carga.

Actividades:

- Diseño de casos de prueba: Crear casos de prueba detallados para cada escenario.
- Ejecución de pruebas: Ejecutar los casos de prueba manualmente y/o automáticamente.
- Registro de defectos: Registrar todos los defectos encontrados y asignarlos a los desarrolladores.
- Verificación de correcciones: Verificar que los defectos se hayan corregido correctamente.

Consideraciones para Pruebas Automatizadas:

Identificar los casos de prueba clave: Priorizar los casos de prueba que se repetirán con frecuencia y que son críticos para la funcionalidad.

Seleccionar la herramienta adecuada: Elegir una herramienta de automatización que se adapte a tu proyecto y a tus conocimientos.

Crear scripts de prueba robustos: Escribir scripts de prueba que sean fáciles de mantener y actualizar.

Casos de Prueba

Pruebas Positivas

Credenciales válidas:

- Usuario y contraseña correctos en mayúsculas y minúsculas.
- Diferentes tipos de contraseñas (simples, complejas, con caracteres especiales).
- Sesiones múltiples:
- Intentar iniciar sesión desde diferentes dispositivos con la misma cuenta.
- Cerrar sesión y volver a iniciar sesión.

Pruebas Negativas:

Credenciales inválidas:

- Usuario o contraseña incorrectos.
- Campos vacíos.
- Contraseña demasiado corta o demasiado larga.
- Caracteres especiales no permitidos.

Condiciones de error:

- Intentos de inicio de sesión fallidos consecutivos.
- Bloqueo de cuenta por demasiados intentos fallidos.

Pruebas de Usabilidad:

- Claridad de los mensajes de error:
- Los mensajes de error son claros y concisos.

Diseño de la interfaz:

- El formulario es intuitivo y fácil de usar.

Pruebas de Seguridad:

Encriptación de contraseñas:

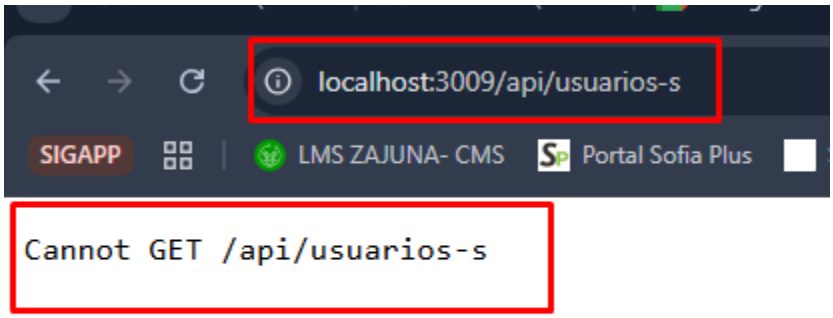
- Las contraseñas se almacenan de forma segura y encriptada.

Casos de Pruebas en POSTMAN

Ejemplo

Caso de Prueba 1: Listar usuario

Descripción	**Listar usuarios del sistema**	
Responsable:	Carlos Castro	
Precondiciones	Tipo de documento identidad, genero, ciudades	
Tipo de caso de prueba	Funcional	
Pasos: 1. Colocar la ruta del recurso de la API (localhost:3009/api/usuarios) **2.** Mostrar listado de dependencias en estado activo		
Resultado esperado	**Resultado real**	**Observación**
Listado de dependencias activas en el sistema	No Cumple con el resultado	Se generar error "Cannot GET /api/usuarios-s"



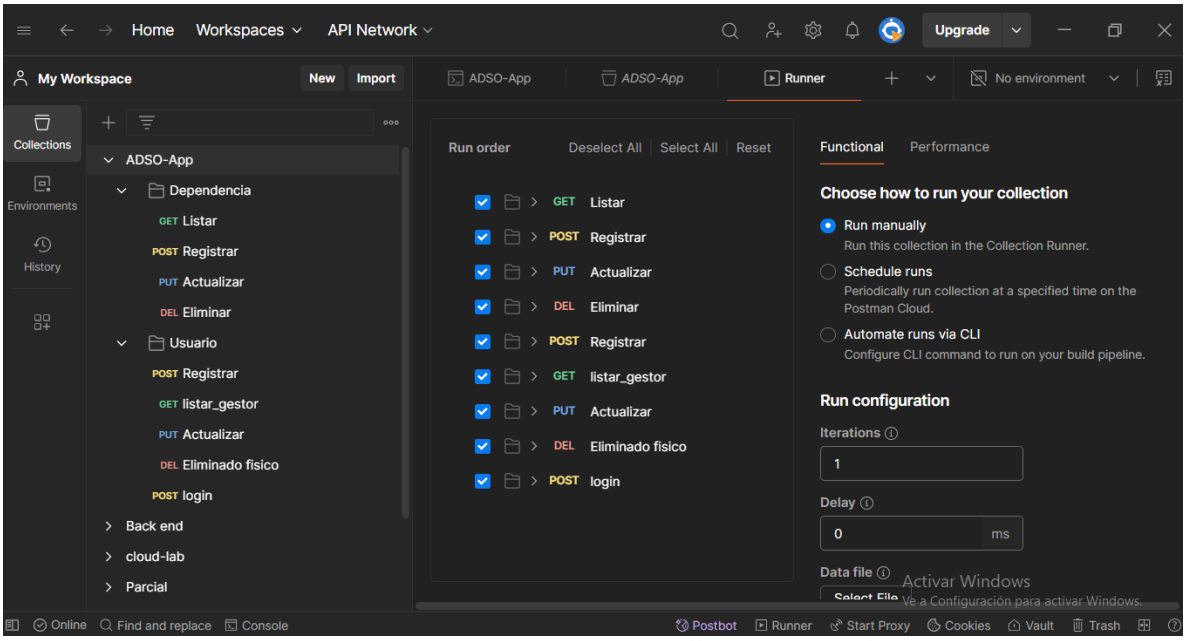
Caso de Prueba 2: Registrar usuario

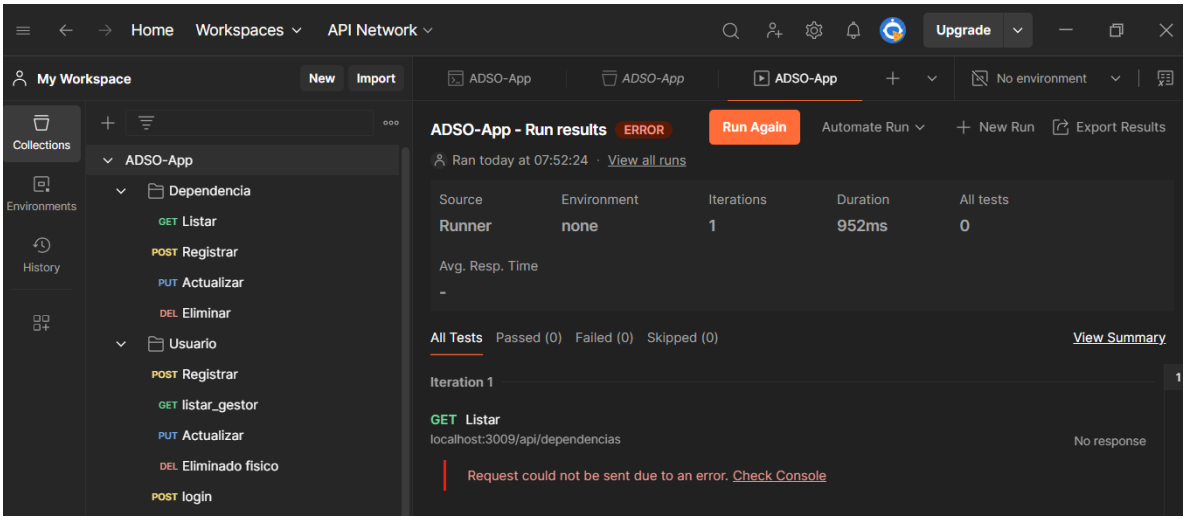
Descripción	**Registrar usuario del sistema**	
Responsable:	Carlos Castro	
Precondiciones	Tipo de documento identidad, país, departamento y ciudad precargado en el sistema.	
Tipo de caso de prueba	Funcional	
Pasos: 1. Colocar la ruta del recurso de la API (localhost:3009/api/usuario) **2.** Diligencia el formulario de la funcionalidad. **3. Dar el en la opción "Send"**		
Resultado esperado	**Resultado real**	**Observación**
Usuario registrado en el sistema	Cumple con el resultado	Aprobado

Ejemplo de un caso de prueba en Postman:

Nombre del caso	Descripción	Método HTTP	URL	Cuerpo de la solicitud	Aserciones
Obtener listado de usuarios	Obtener información de los usuarios existentes	GET	/usuarios		Código de estado 200, cuerpo de la respuesta contiene el usuario
Crear usuario	Registrar un nuevo usuario	POST	/usuarios	Datos del usuario	Código de estado 201, cuerpo de la respuesta
Eliminar usuario	Eliminar usuario del sistema	DELETE	/usuario	Identificador del usuario	Código de estado 201, cuerpo de la respuesta
Actualizar usuario	Actualizar los datos del usuario	PUT	/usuario	Datos del usuario	Código de estado 200, cuerpo de la respuesta

Plan de pruebas en POSTMAN





Procesos de calidad del software

Aseguramiento de la Calidad (QA):

- Revisión de código: Explica cómo se realiza la revisión del código para identificar posibles errores y mejoras.
- Pruebas estáticas: Detalla las herramientas y técnicas utilizadas para analizar el código sin ejecutarlo (análisis de flujo de control, análisis de complejidad, etc.).
- Pruebas dinámicas: Describe las diferentes tipos de pruebas realizadas (unitarias, de integración, funcionales, de rendimiento, etc.) y las herramientas utilizadas.
- Metricas de calidad: Indica qué métricas se utilizan para medir la calidad del código (cobertura de código, complejidad ciclomática, etc.).

Control de calidad:

- Plan de pruebas: Explica cómo se ha elaborado el plan de pruebas, incluyendo los casos de prueba, los criterios de aceptación y los recursos necesarios.
- Gestión de defectos: Describe el proceso de seguimiento y resolución de defectos encontrados durante las pruebas.
- Informes de pruebas: Explica cómo se generan los informes de pruebas y qué información contienen.

Mejora continua:

- Análisis de resultados: Describe cómo se analizan los resultados de las pruebas para identificar áreas de mejora.
- Acciones correctivas: Explica las acciones que se toman para corregir los defectos y mejorar la calidad del software.
- Prevención de defectos: Describe las medidas implementadas para prevenir la aparición de nuevos defectos.

Herramientas y tecnologías:

- Herramientas de gestión de calidad: Menciona las herramientas utilizadas para gestionar el ciclo de vida de los defectos (Jira, Bugzilla, etc.).
- Herramientas de automatización de pruebas: Describe las herramientas utilizadas para automatizar las pruebas (Selenium, Postman, etc.).

Estándares de calidad:

- Normas y estándares: Indica si se han seguido alguna norma o estándar de calidad específico (ISO 9126, CMMI, etc.).
- Mejores prácticas: Describe las mejores prácticas de desarrollo de software que se han aplicado.

Bibliografía

Para desarrollar este plan de pruebas, se han considerado conceptos y buenas prácticas de las siguientes fuentes abiertas de información sobre pruebas de software y gestión de casos de prueba:

- ISTQB (International Software Testing Qualifications Board) - Una guía estándar que detalla métodos de prueba de software y casos de uso.
- Guru99 Software Testing - Un recurso educativo sobre las mejores prácticas y estrategias para pruebas manuales y automatizadas.
- Mozilla Open Access - Documentación extensa sobre control de calidad y aseguramiento de software en aplicaciones web.

Caso de Prueba 1: Login con Credenciales Válidas

Descripción	Verificar que un usuario registrado puede iniciar sesión correctamente	
Responsable:	Carlos Castro	
Precondiciones	El usuario ya está registrado y tiene una cuenta verificada	
Tipo de caso de prueba	Funcional, automatizada y usabilidad	
Pasos:		
Resultado esperado	**Resultado real**	**Observación**
1. Navegar a la página de inicio de sesión. 2. Ingresar número de documento de identidad y contraseña correctos. 3. Hacer clic en el botón "Iniciar sesión". 4. Ingresar a la sesión del usuario. 5. Acceso exitoso a la cuenta del usuario	**Ok** **Ok** La etiqueta del botón muestra el texto “Ingresar”, ver imagen 1 Ok Ok	
Resultado esperado: El usuario es redirigido a la página principal con un mensaje de bienvenida.		