EVALUACIÓN

EVALUACIÓN DE SISTEMAS QA
Semana 3

Michel Brevis
06-11-2024
Técnico en Análisis y Programación Computacional

DESARROLLO:

1. **¿Cuáles son las ventajas de generar informes sobre los resultados de las pruebas automatizadas en el contexto de la aplicación de cálculo de fuerza? Además, proporciona un modelo de informe (ya sea en forma de documento o tabla) que brinde una visión de los resultados de las pruebas realizadas en la aplicación.**

Generar informes sobre los resultados de las pruebas automatizadas en el contexto de una aplicación de cálculo de fuerza tiene varias ventajas:

Identificación Rápida de Errores:
Los informes de pruebas automatizadas ayudan a identificar rápidamente problemas en la aplicación, como en este caso, donde el sistema calcula un resultado positivo cuando la aceleración es negativa. Esto permite a los desarrolladores detectar errores de lógica o implementación que de otro modo podrían pasar desapercibidos.

Mejora en la Calidad del Software:
Los informes detallados permiten mantener un registro de los casos fallidos y las áreas problemáticas, facilitando la mejora continua del software. Al analizar estos informes, el equipo puede tomar medidas correctivas para mejorar la precisión y confiabilidad de la aplicación.

Evidencia Documentada para el Equipo y los Clientes:
Los informes de pruebas son útiles para demostrar a los clientes y stakeholders que el software ha sido evaluado rigurosamente y que cualquier problema identificado está siendo abordado. Esto genera confianza y transparencia en el proceso de desarrollo.

Facilita la Revisión y el Análisis:
Los informes permiten que otros miembros del equipo revisen los resultados y contribuyan con ideas o soluciones. En el caso de un fallo, como el mostrado en el reporte, los desarrolladores pueden discutir la causa y realizar ajustes en la lógica del sistema.

Ahorro de Tiempo en el Ciclo de Desarrollo:
Al automatizar las pruebas y generar informes, se reduce el tiempo necesario para realizar pruebas manuales y verificar los resultados. Esto permite que el equipo de desarrollo se concentre en otras tareas y que las pruebas sean más rápidas y consistentes.

Historial de Pruebas para Evaluaciones Futuras:
Tener un registro histórico de los resultados permite analizar la evolución del software a lo largo del tiempo. Si un error vuelve a ocurrir, el equipo puede revisar informes pasados para ver cómo se resolvieron problemas similares anteriormente.

Cumplimiento de Estándares y Buenas Prácticas:
La generación de informes de pruebas también ayuda a cumplir con los estándares de calidad y buenas prácticas en el desarrollo de software. Permite a los equipos de QA (Quality Assurance) asegurar que la aplicación cumpla con los requisitos funcionales establecidos.

Al momento de generar el informe, genero el siguiente modelo para el registro y seguimiento de errores.

PLANTILLA DE INFORME DE ERRORES

ID DE DEFECTO	DESCRIPCIÓN	SEVERIDAD	IMPACTO EN EL NEGOCIO	FUNCIONALIDAD	RENDIMIENTO	ESTABILIDAD	GRÁFICO / DETALLES DE UX
		Showstopper					
		Destacado					
		Menor					
		Bajo					

2. **Como sugerencia, ¿qué estrategias podrían implementarse para mejorar la calidad de las pruebas en la aplicación de cálculo de fuerza?**

Para mejorar la calidad de las pruebas en la aplicación de cálculo de fuerza, se pueden implementar las siguientes estrategias:

Ampliar los Casos de Prueba con Valores Extremos y Borde:

Realizar pruebas con valores de masa y aceleración extremadamente altos o bajos, incluyendo cero y valores negativos para ambos parámetros. Esto ayuda a validar cómo maneja la aplicación situaciones poco comunes o límites.

Incluir casos en los que uno de los valores sea cero para verificar que el resultado sea cero, cumpliendo con las leyes físicas.

Pruebas de Validación de Entrada:

Asegurarse de que la aplicación valide correctamente las entradas del usuario. Implementar pruebas para entradas no válidas, como caracteres especiales, texto en lugar de números o valores fuera de los rangos permitidos.

Incluir mensajes de error claros y específicos para las entradas inválidas para mejorar la experiencia del usuario y evitar cálculos incorrectos.

Pruebas de Precisión y Redondeo:

Validar que la aplicación maneje correctamente los decimales y el redondeo de resultados, especialmente en valores pequeños que pueden generar errores de precisión.

Asegurarse de que el sistema redondee correctamente según los requisitos establecidos (por ejemplo, a dos decimales) para evitar inconsistencias en los cálculos.

Pruebas Automatizadas de Regresión:

Implementar un conjunto de pruebas automatizadas de regresión que se ejecuten cada vez que se realizan cambios en la aplicación. Esto garantiza que nuevas modificaciones no introduzcan errores en funciones previamente validadas.

Estas pruebas pueden incluir todos los casos de prueba críticos para la aplicación, asegurando que los cálculos de fuerza siempre sean correctos tras cada actualización.

Incluir Pruebas de Rendimiento:

Evaluar cómo se comporta la aplicación bajo cargas pesadas o en dispositivos de bajo rendimiento. Por ejemplo, ingresando y calculando rápidamente múltiples valores para verificar si la aplicación responde de manera eficiente.

Esto es especialmente útil si la aplicación se utilizará en dispositivos con diferentes capacidades de hardware.

Pruebas de Usabilidad:

Asegurarse de que la interfaz sea clara y fácil de usar para que los usuarios puedan ingresar datos y entender los resultados sin problemas.
Incluir pruebas que evalúen si los usuarios comprenden correctamente cómo ingresar los valores de masa y aceleración, y cómo interpretar el resultado de la fuerza calculada.

Pruebas de Integración con Otros Módulos (si aplica):

Si la aplicación de cálculo de fuerza se integra con otros módulos o servicios, como bases de datos o servicios de autenticación, se deben realizar pruebas de integración para asegurar que funcione correctamente en todos los flujos.

Pruebas de Seguridad:

Implementar pruebas para verificar que el sistema maneje de manera segura los datos de entrada, evitando problemas como la inyección de scripts o la manipulación de datos.
Esto es especialmente importante si el sistema se implementa en un entorno web donde los datos del usuario pueden estar expuestos a posibles ataques.

Revisión de Código y Auditoría Técnica:

Realizar revisiones periódicas de código por parte de otros desarrolladores para identificar posibles errores lógicos o mejoras en la implementación.
Las auditorías técnicas ayudan a garantizar que el código esté alineado con las mejores prácticas de programación y eficiencia.

Pruebas de Compatibilidad en Diferentes Navegadores y Dispositivos:

Probar la aplicación en distintos navegadores y sistemas operativos (como Chrome, Firefox, Edge, Windows, MacOS, y Linux) para garantizar que funcione correctamente en todos los entornos soportados.

3. **3. ¿Cuál es la función de las herramientas de gestión de reportes de defectos? Utilizando Jira, registra el defecto correspondiente al caso de prueba fallido de la aplicación de fuerza, documenta la posible corrección y verifica la solución. A continuación, incluye en el documento las capturas de pantalla que ilustren cada paso.**

Las herramientas de gestión de reportes de defectos, también conocidas como herramientas de *bug tracking* o *issue tracking*, cumplen un papel fundamental en el proceso de desarrollo y control de calidad del software. Su función principal es permitir el registro, seguimiento, y gestión de defectos o errores en el sistema, desde su identificación hasta su resolución.

Registro de Defectos:
Permiten registrar los detalles completos de cada defecto, como la descripción, la gravedad, la prioridad, el ambiente en el que ocurrió y los pasos necesarios para reproducirlo. Esta información ayuda a los desarrolladores a entender el contexto del problema.

Seguimiento del Ciclo de Vida de los Defectos:
Las herramientas permiten gestionar el ciclo de vida de un defecto, desde su reporte hasta su resolución. Los defectos pasan por diferentes estados (como "nuevo", "en progreso", "resuelto", "cerrado", etc.), y la herramienta facilita el seguimiento de estos cambios, asegurando que cada defecto sea atendido en el momento adecuado.

Asignación y Priorización de Defectos:
Permiten asignar defectos a miembros específicos del equipo, como desarrolladores o testers, y establecer la prioridad y la severidad del defecto. Esto asegura que los defectos críticos se aborden con rapidez y que los recursos se asignen eficientemente.

Comunicación y Colaboración entre Equipos:
Las herramientas de gestión de defectos facilitan la comunicación entre diferentes equipos, como el de desarrollo, pruebas, y gestión de proyectos. Todos los miembros del equipo pueden acceder a la misma información sobre el estado y los detalles de un defecto, lo que mejora la colaboración y evita la duplicación de esfuerzos.

Análisis de Tendencias y Generación de Informes:
Estas herramientas permiten generar informes detallados sobre los defectos encontrados y su resolución. Esto incluye métricas como el número de defectos encontrados, resueltos y pendientes, el tiempo promedio de resolución, y los módulos del sistema con más errores. Estos informes ayudan a identificar patrones de calidad, áreas problemáticas y permiten tomar decisiones para mejorar el proceso de desarrollo.

Historial y Trazabilidad:
Mantienen un historial completo de cada defecto, lo que permite realizar un seguimiento de los cambios, versiones, y cualquier comentario o anotación hecha sobre el defecto. Esta trazabilidad es útil para auditorías y análisis posteriores, ya que permite revisar cómo se manejaron los defectos en el tiempo.

Integración con Otras Herramientas de Desarrollo:

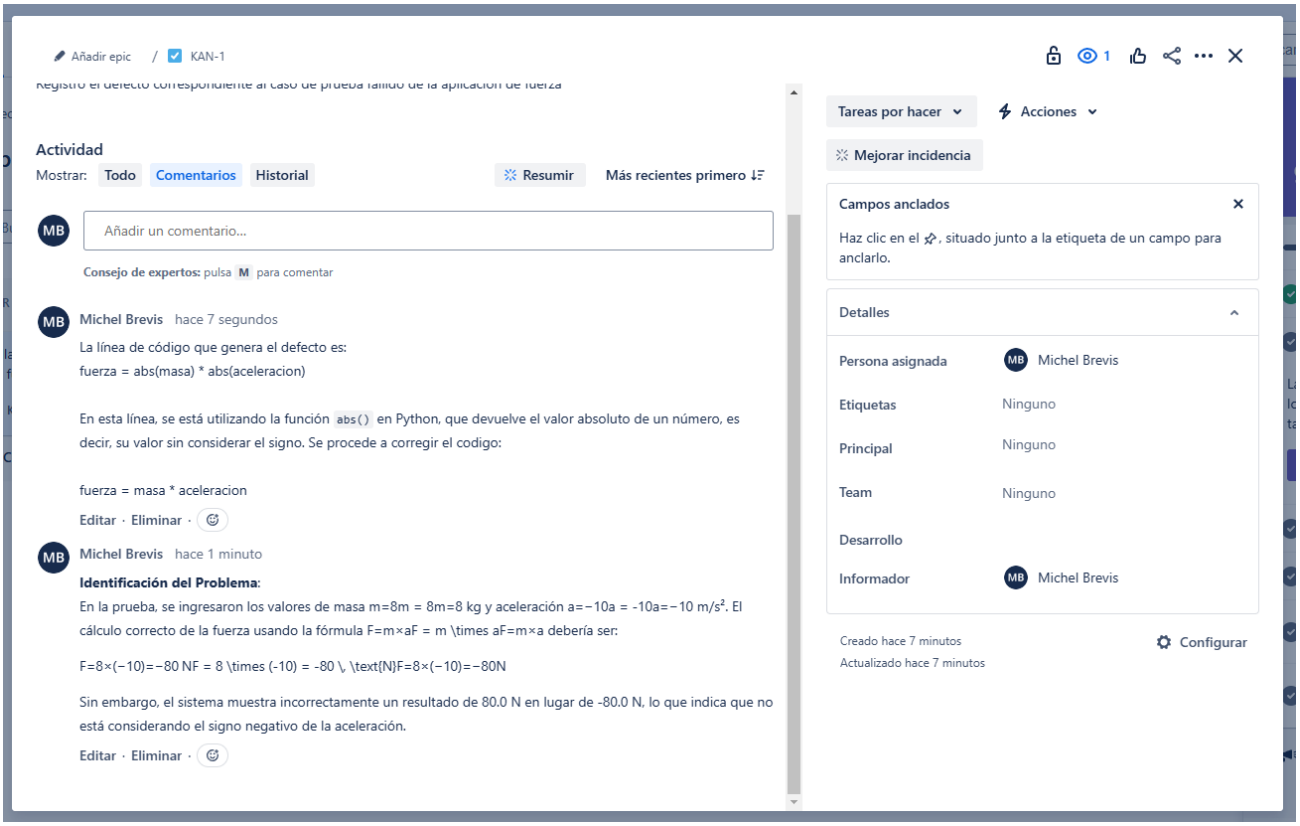
Muchas herramientas de gestión de defectos se integran con otras herramientas de desarrollo, como sistemas de control de versiones (Git, SVN), herramientas de pruebas automatizadas, y sistemas de integración continua. Esto permite un flujo de trabajo más cohesivo y automatizado, donde los defectos pueden ser registrados automáticamente tras una prueba fallida, por ejemplo.

Priorizar la Mejora Continua:

Al almacenar datos históricos sobre los defectos, estas herramientas ayudan a las organizaciones a realizar un análisis retrospectivo y a priorizar áreas que requieren mejoras en términos de calidad y estabilidad. Esto contribuye a la mejora continua de los procesos de desarrollo y pruebas.

Algunos ejemplos populares de herramientas de gestión de defectos incluyen **JIRA**, **Bugzilla**, **Trello** (cuando se configura para el seguimiento de defectos), y **Redmine**. Estas herramientas son esenciales en equipos de desarrollo modernos, ya que centralizan la gestión de defectos, agilizan la resolución de problemas, y contribuyen a mantener altos estándares de calidad en el software.

Para la documentación del informe de error en el caso fallido de la aplicación de fuerza, se usará jira.



REFERENCIAS BIBLIOGRÁFICAS:

- **Ejemplo texto de lectura de IACC:**

IACC. (2024). *Evaluación de sistemas QA* Semana 3