Informe de Testing y Pruebas de Código

Introducción

- El testing y las pruebas de código desempeñan un papel crucial en la identificación de errores y lograr el funcionamiento adecuado. Esta práctica contribuye a una calidad y robustez altos en los productos finales, reduciendo también costes asociados a fallos en etapas posteriores.

Conceptos Básicos

- **Definición y Diferencia entre Testing y Pruebas de Código**

    - **Testing:** Esta práctica contribuye de manera significativa a alcanzar una calidad y robustez altas en los productos finales, al tiempo que reduce los costes asociados a fallos en etapas posteriores del desarrollo. En este contexto, es de gran importancia comprender los conceptos básicos relacionados con el testing y las pruebas de código, así como los beneficios asociados a su implementación.

    - **Pruebas de Código:** Las pruebas de código se centran en verificar el comportamiento y la calidad del código, lo que incluye la revisión de normas, las pruebas unitarias, las pruebas de integración y otras prácticas relacionadas. Ambas actividades son fundamentales para garantizar la fiabilidad y eficacia de los productos de software. (Pittet, 2024)

Ilustración 1 Bing. (2024) https://www.bing.com/images/create/una-imagen-que-ilustre-la-definicic3b3n-y-diferencia-/1-65acd6994c874ddc86ffb4297006a178?id=iE99hTdVBxdGZ%2FbJyL9Zlw%3D%3D&view=detailv2&idpp=genimg&idpclose=1&form=SYDBIC

- **Objetivos y Beneficios de Realizar Pruebas**

    - **Objetivos:**
        - Asegurar la calidad de software.
        - Identificar y corregir errores.
        - Aumentar la confianza del producto

    - **Beneficios:**
        - Reducción de costos en la corrección de errores.
        - Reducción en los tiempos de desarrollo.
        - Confiabilidad y robustez del software.

Tipos de Pruebas

[+] Unitarias

- Pruebas para verificar la funcionalidad de unidades individuales del código. ***Herramientas:*** *Junit, NUnit.*

[+] Integración

- Asegura que diferentes componentes del sistema funcionen correctamente de forma conjunta. ***Herramientas:*** *Selenium, TestNG.*

[+] Sistema

- Verifica el sistema en conjunto para garantizar que cumple con los requisitos. ***Herramientas:*** *Selenium, Appium.*

[+] Aceptación

- Evalúa si el software cumple con los requisitos establecidos por el cliente. ***Herramientas:*** *Cucumber, SpecFlow.*

[+] Carga y Estrés

- Evalúa el rendimiento bajo, medio y alto. ***Herramientas****: Apache JMeter, Gatling.* (loadview, 2024)

Técnicas de Testing

- **TDD (Test Driven Develpoment)**
    - Consiste en desarrollar pruebas antes de escribir el código. Los beneficios son la identificación temprana de problemas.

- **BDD (Behavior Driven Development)**
    - Enfocado en el comportamiento del software utilizando un lenguaje cercano. ***Ejemplo:*** *Cucumber.* (John, 2024)

Automatización de Pruebas

Centrado en acelerar el proceso de testing, garantizando de esta forma, una rápida ejecución y consistente.

Herramientas y Frameworks Populares

- **Selenium** *-Para pruebas de Interfaz web.*
- **Junit y TestNG** *-Para pruebas unitarias e integración.*
- **Appium** *-Automatización de pruebas móviles.*
- **Cypress** *-Pruebas de extremo a extremo.*
- **JMeter** *-Pruebas de carga y rendimiento.* (Valls, 2024)

Casos de Uso y Ejemplos

Caso de Uso 1: Pruebas Unitarias en Desarrollo de Software Empresarial

En un proyecto empresarial de desarrollo de software, la implementación de pruebas unitarias es esencial para validar la funcionalidad individual de componentes o módulos. En este contexto, consideremos un sistema de gestión de recursos humanos. Al aplicar pruebas unitarias, se pueden verificar funciones específicas, como el cálculo de salarios o la actualización de registros. Estas pruebas aseguran que cada unidad de código opere correctamente, lo que contribuye a la identificación temprana de posibles errores en funcionalidades clave. Además, las pruebas unitarias facilitan la detección de problemas en una etapa temprana del desarrollo, lo que conduce a un código más robusto y de mayor calidad.

Caso de Uso 2: Pruebas de Carga en Aplicaciones Web

Para proyectos web de gran envergadura, las pruebas de carga son esenciales. Supongamos una plataforma de streaming de video. Al simular un gran número de usuarios concurrentes, las pruebas de carga permiten evaluar la capacidad del sistema para manejar la carga esperada. Identificar posibles cuellos de botella y optimizar el rendimiento se vuelve crucial en este escenario. La importancia radica en garantizar que la aplicación pueda proporcionar una experiencia de usuario fluida incluso bajo condiciones de alto tráfico. Sin estas pruebas, podrían surgir problemas como la degradación del rendimiento o interrupciones inesperadas, afectando negativamente la satisfacción del usuario. Las pruebas de carga proporcionan una visión valiosa sobre cómo se comportará la aplicación en situaciones de uso intensivo y permiten tomar medidas preventivas para mejorar su escalabilidad.

Conclusión

En conclusión, son fundamentales el testing y las pruebas de código para garantizar la calidad y robustez del software. En conjunto, los diferentes tipos de pruebas, técnicas de testing y la automatización contribuyen en un desarrollo eficiente y a los productos de confiabilidad. Es clave la inversión en pruebas adecuadas desde el principio del desarrollo ya que resulta en beneficios significativos, tanto en términos de calidad del producto como el ahorro de costos a largo plazo.

Bibliografía

John, S. (20 de 01 de 2024). *CUELOGIC*. Obtenido de https://www.cuelogic.com/blog/bdd-vs-tdd#:~:text=BDD%20is%20meant%20to%20test%20how%20an%20application,BDD%20with%20the%20outcome%20of%20more%20complex%20scenarios.

loadview. (20 de 01 de 2024). Obtenido de https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/

Pittet, S. (20 de 01 de 2024). *ATLASSIAN*. Obtenido de https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing

Valls, M. (20 de 01 de 2024). *cleverit*. Obtenido de https://www.cleveritgroup.com/blog/5-frameworks-comunes-en-automatizacion-de-pruebas-y-cuando-usarlos