Testing en el mundo de desarrollo de software.

* Introducción
* Tipo de Testing y sus importancias
* Anti padrones y Errores Comunes
* Integración continua
* Resumen

Introducción

* ¿Qué es calidad?

Tipo de Testing y sus importancias

* Unitarias
* Integración
  + ¿Porque automatizar pruebas?
  + ¿Qué automatizar y que no automatizar?
* Aceptación de Usuarios o Funcionales
* Desempeños

Recuadro de Tipos de Test que debemos hacer.

“Hacer las pruebas son **Difíciles**, pero no hacerlas es aun más”

Calidad: El estándar de algo medido contra otras cosas de un tipo similar; El grado de excelencia de algo.

**UNIT TESTING** es un nivel de prueba de software en el que se prueban unidades / componentes individuales de un software. El propósito es validar que cada unidad del software funcione según lo diseñado. Una unidad es la parte comprobable más pequeña de cualquier software. Por lo general, tiene una o algunas entradas y, por lo general, una sola salida.

**Ventajas de las pruebas unitarias**

Las pruebas unitarias aumentan la confianza en cambiar / mantener el código. Si se escriben buenas pruebas unitarias y se ejecutan cada vez que se cambia un código, podremos detectar de inmediato cualquier defecto introducido debido al cambio. Además, si los códigos ya son menos interdependientes para hacer posibles las pruebas unitarias, el impacto involuntario de los cambios en cualquier código es menor.

Los códigos son más reutilizables. Para que las pruebas unitarias sean posibles, los códigos deben ser modulares. Esto significa que los códigos son más fáciles de reutilizar.

El desarrollo es más rápido. ¿Cómo? Si no tiene una prueba de unidad en su lugar, escriba su código y realice esa 'prueba de desarrollador' difusa (establece algunos puntos de interrupción, activa la GUI, proporciona algunas entradas que esperamos golpeen su código y esperan que esté todo configurado. ) Pero, si tiene pruebas unitarias, escribe la prueba, escribe el código y ejecuta la prueba. Escribir pruebas lleva tiempo, pero el tiempo se compensa con la menor cantidad de tiempo que lleva ejecutar las pruebas; No necesita iniciar la GUI y proporcionar todas esas entradas. Y, por supuesto, las pruebas unitarias son más confiables que las 'pruebas de desarrollador'. El desarrollo también es más rápido a largo plazo. ¿Cómo? El esfuerzo requerido para encontrar y corregir los defectos encontrados durante las pruebas unitarias es muy menor en comparación con el esfuerzo requerido para reparar los defectos encontrados durante las pruebas del sistema o las pruebas de aceptación.

El costo de reparar un defecto detectado durante las pruebas unitarias es menor en comparación con el de los defectos detectados en niveles superiores. Compare el costo (tiempo, esfuerzo, destrucción, humillación) de un defecto detectado durante las pruebas de aceptación o cuando el software está activo.

La depuración es fácil. Cuando falla una prueba, solo se deben depurar los últimos cambios. Con las pruebas a niveles más altos, los cambios realizados en el transcurso de varios días / semanas / meses deben ser escaneados.

Los códigos son más confiables. ¿Por qué? Creo que no hay necesidad de explicar esto a una persona cuerda.

**Pruebas de integración**

Es un nivel de prueba de software donde las unidades individuales se combinan y se prueban como un grupo. El propósito de este nivel de prueba es exponer fallas en la interacción entre unidades integradas. Los controladores de prueba y los apéndices de prueba se utilizan para ayudar en las pruebas de integración.

**SYSTEM TESTING**

**E**s un nivel de prueba de software en el que se prueba un software completo e integrado. El propósito de esta prueba es evaluar el cumplimiento del sistema con los requisitos especificados.

El servicio de Pruebas de Desempeño permite evaluar el comportamiento de una aplicación ante condiciones extremas tales como picos de usuarios utilizándolo simultáneamente y, con base en esta información, realizar ajustes para mejorar su rendimiento.

**Pruebas de aceptación**

Pruebas formales con respecto a las necesidades del usuario, los requisitos y los procesos comerciales llevados a cabo para determinar si un sistema cumple o no con los criterios de aceptación y para permitir que el usuario, los clientes u otra entidad autorizada determinen si aceptan o no el sistema, Por lo general, el método de prueba de caja negra se usa en las pruebas de aceptación

**Selenium WebDriver**

* Objetivos
* Para que automatizar

**Ejercicio:**



1. Cómo descargar e instalar Selenium WebDriver y Primer script Selenium Webdriver: ejemplo de código JAVA
2. Localizadores en Selenium IDE: (<https://www.guru99.com/locators-in-selenium-ide.html>)

* Selector CSS
* DOM
* Texto de enlace
* ID

1. Encuentra Element y FindElements en Selenium WebDriver y Selenium Form WebElement: TextBox, botón Enviar, sendkeys (), clic () y Cómo seleccionar CheckBox y botón de radio en Selenium WebDriver. (<https://www.guru99.com/find-element-selenium.html>)
2. Localizadores en Selenium IDE 2: (<https://www.guru99.com/locators-in-selenium-ide.html>)

* XPath
  + XPath en Selenium WebDriver: Tutorial completo, XPath contiene, hermano, funciones ancestrales en Selenium WebDriver
  + <https://www.guru99.com/xpath-selenium.html>
  + <https://www.guru99.com/using-contains-sbiling-ancestor-to-find-element-in-selenium.html>

1. Page Object Model (POM) y Page Factory: Tutorial de Selenium WebDriver y Espera implícita, explícita y fluida en Selenium WebDriver (<https://www.guru99.com/page-object-model-pom-page-factory-in-selenium-ultimate-guide.html>) y Muestra de **Selenium IDE**