PORTADA

Nombre Alumno / DNI	**Hugo Carvajal / 5048895T**
Título del Programa	**1ºPHE CYBERSECURITY & DIGITAL INTELLIGENCE**
Nº Unidad y Título	**UNIT 1-PROGRAMMING & CODING**
Año académico	**2023-2024**
Profesor de la unidad	**Gabriela García**
Título del Assignment	**AB FINAL**
Día de emisión	**13 de octubre de 2023**
Día de entrega	**Fecha en la que entrega el alumno esté AB Final. 30/01/2024**
Nombre IV y fecha	
Declaración del estudiante	**Certifico que la presentación del assignment es completamente mi propio trabajo y entiendo completamente las consecuencias del plagio.** **Entiendo que hacer una declaración falsa es una forma de mala práctica.** **Fecha: 30/01/2024** **Firma del alumno: Hugo Carvajal**

Plagio

El plagio es una forma particular de hacer trampa. El plagio debe evitarse a toda costa y los alumnos que infrinjan las reglas, aunque sea inocentemente, pueden ser sancionados. Es su responsabilidad asegurarse de comprender las prácticas de referencia correctas. Como alumno de nivel universitario, se espera que utilice las referencias adecuadas en todo momento y mantenga notas cuidadosamente

detalladas de todas sus fuentes de materiales para el material que ha utilizado en su trabajo, incluido

cualquier material descargado de Internet. Consulte al profesor de la unidad correspondiente o al tutor del curso si necesita más consejos.

MSMK University

ÍNDICE

Introducción:

Breve descripción sobre la relevancia del testing y pruebas de código en el ciclo de vida del desarrollo de software.

La prueba de software es probablemente la parte menos comprendida del ciclo de vida del desarrollo de software. En este trabajo, mediante distintas fases organizadas metodológicamente, se garantiza la calidad, la confiabilidad y rendimiento del producto final.

Conceptos Básicos:

Definición y diferencia entre testing y pruebas de código.

TESTING: El testing, es un proceso para verificar y validar la funcionalidad de un programa o una aplicación de software con el objetivo de garantizar que el producto de software esté libre de defectos.

PRUEBAS DE CÓDIGO: Las pruebas de código son una parte específica del proceso de testing y se centran en evaluar las unidades individuales de código fuente para verificar su correcto funcionamiento.

DIFERENCIA: La diferencia es que las pruebas de código es una parte específica del testing como he explicado anteriormente y el testing se encarga de la evaluación general del software.

Objetivos y beneficios de realizar pruebas.

Objetivos de realizar las pruebas:

Asegurar la Estabilidad, Identificar Errores, Optimizar el Rendimiento, Validar Requisitos, Mejorar la Calidad.

Encontrar el mayor número de defectos en el código para que se resuelvan y eliminarlos.

Asegurar que el producto funciona tal y cómo se ha definido en los requisitos.

Dar información al cliente de los defectos que no se han podido eliminar del producto final.

Proporcionar al producto final una mayor de calidad.

Beneficios:

Reducción de Errores en Producción, Mejora de la Confianza del Usuario, Cumplimiento de Requisitos, Ahorro de Costos.

La calidad de código mejora puesto que podemos detectar errores en una etapa más temprana de desarrollo y de forma más rápida.

Puedes trabajar de una forma más ágil, ya que facilita los cambios y favorece la integración.

Los propios test pueden funcionar como documentación y ejemplos de nuestras clases.

Reduce el costo de mantenimiento del proyecto.

A través del desarrollo guiado por pruebas, TDD por sus siglas en inglés, nos ayuda a mejorar el diseño de nuestro software.

Tipos de Pruebas:

Descripción detallada de diferentes tipos de pruebas.

Herramientas populares asociadas a cada tipo de prueba.

Pruebas Unitarias

Descripción: Es una forma efectiva de comprobar el correcto funcionamiento de las unidades individuales más pequeñas de los programas informáticos.

Herramientas Populares:

JUnit (Java)

NUnit (C#)

PHPUnit (PHP)

Pruebas de Integración

Descripción: Las pruebas de integración se definen como un mecanismo de testeo de software, donde se realiza un análisis de los procesos relacionados con el ensamblaje o unión de los componentes, sus comportamientos con múltiples partes del sistema o de hardware, entre otras.

Herramientas Populares:

Apache JMeter (para pruebas de integración de carga)

TestNG (Java)

Pruebas Funcionales:

Descripción: Las pruebas funcionales en las pruebas de software son una forma de determinar si el software o una aplicación funcionan como se espera. Las pruebas funcionales no se ocupan de cómo se produce el procesamiento, sino de si éste ofrece los resultados correctos o tiene algún fallo.

Herramientas Populares:

Selenium (para pruebas de interfaz de usuario)

Appium (para pruebas de aplicaciones móviles)

Pruebas de Regresión:

Descripción: La prueba de regresión es un método común que la mayoría de los procesos de desarrollo de software utilizan para verificar la funcionalidad del software después de realizar modificaciones en él.

Herramientas Populares:

TestNG (Java)

pytest (Python)

Pruebas de Rendimiento:

Descripción: Las pruebas de rendimiento son un proceso que determina si un software cumple con los requisitos de velocidad, escalabilidad y estabilidad bajo diferentes cargas de trabajo. El objetivo principal es identificar y eliminar cuellos de botella donde la aplicación puede fallar o presentar demoras.

Herramientas Populares:

LoadRunner

Gatling

Pruebas de Aceptación:

Descripción: La prueba de aceptación del usuario, también conocida como prueba beta, se define como la prueba del software por parte del usuario o cliente para determinar si puede ser aceptado o no.

Herramientas Populares:

No hay herramientas predeterminadas, las pruebas suelen realizarse con la prueba del producto de usuarios finales.

Técnicas de Testing:

Exploración de técnicas como TDD (Test Driven Development), BDD (Behavior Driven Development) y otras relevantes.

Beneficios y retos asociados a cada técnica.

TDD: TDD o Test-Driven Development es una práctica de programación que consiste en escribir primero las pruebas (generalmente unitarias), después escribir el código fuente que pase la prueba satisfactoriamente y, por último, refactorizar el código escrito.

Beneficios:

- Usando procesos TDD evitamos escribir codigo innecesario.
- Desarrollo ágil.
- Mejor integración con terceros.
- Feedback de API
- Depuración

Retos:

- Puede requerir un cambio en la mentalidad del desarrollador.
- Puede parecer lento porque hay que escribir mucha documentación de las pruebas.

BDD: BDD es Behavior Driven Development, o lo que es lo mismo en español, desarrollo guiado por comportamiento. Es un proceso de software ágil que busca la colaboración y entendimiento entre desarrolladores, gestores de proyecto y equipo de negocio.

Beneficios:

- Fuerte colaboración.
- Alta visibilidad.
- El diseño del software sigue el valor comercial.
- El lenguaje omnipresente.
- El desarrollo de software cumple con la necesidad del usuario.
- Más confianza del lado del desarrollador.
- Costes más bajos.

Retos:

- Elegir las Herramientas y Tecnologías adecuadas
- Mantener la coherencia entre escenarios y código
- Percepción de un aumento inicial en la carga de trabajo

Automatización de Pruebas:

Introducción a la automatización y sus ventajas.

Se conoce como automatización de pruebas al proceso de realizar pruebas o test de manera automática, gestionando esos datos para poder aumentar la calidad del software desarrollado utilizando herramientas y scripts para realizar tareas repetitivas.

Ventajas:

- Mayor precisión
- Reducción de tiempo y costo
- Mayor reutilización
- Incrementa la productividad de los equipos de trabajo

Herramientas y frameworks populares para la automatización de pruebas.

Casos de Uso y Ejemplos:

- **Selenium:** Es un entorno de pruebas de software para aplicaciones basadas en la web. Selenium provee una herramienta de grabar/reproducir para crear pruebas sin usar un lenguaje de scripting para pruebas
- **Protactor:** Es la herramienta que la gente de Google nos ofrece para la implementación de pruebas E2E de aplicaciones AngularJS, y recalco lo de aplicaciones AngularJS porque solo funciona con este tipo de aplicaciones
- **Appium:** Es una herramienta de automatización de código abierto para ejecutar scripts y probar aplicaciones nativas, aplicaciones web y aplicaciones híbridas sobre Android o iOS utilizando un webdriver.
- **JUnit:** Es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java. JUnit es un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada.

Presenta algunos ejemplos prácticos o casos de uso donde se aplican distintos tipos de pruebas y técnicas de testing en proyectos reales.

Prueba Unitaria: Creamos pruebas unitarias que nos calculan diferentes parámetros de una cuenta bancaria como puede ser el saldo.

Prueba Funcional: Creamos un sistema para que podamos reservar habitaciones para hoteles en línea y realizamos pruebas para comprobar que posibles clientes puedan realizar reservas, cancelar reservas y demás pruebas

Conclusión:

Reflexión sobre la importancia de las pruebas y el testing en garantizar la calidad y confiabilidad del software.

Para mi son muy importante las pruebas y el testing ya que son muy importantes a la hora de darnos una buena calidad y confiabilidad del código y nos puede entre otras cosas ahorrar mucho trabajo a largo tiempo y proporcionarnos documentación importante sobre el código.

REFERENCIAS

INTRODUCCION

Avaible at: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0121-49932011000300006

CONCEPTOS BÁSICOS:

Avaible at: https://profile.es/blog/que-es-el-testing-de-software/

OBJETIVOS Y BENEFICIOS DE REALIZAR PRUEBAS:

Avaible at: https://testerhouse.com/teoria-testing/objetivo-de-las-pruebas-de-software/

https://platzi.com/blog/testing-ventajas-formas-de-realizar-pruebas/

DESCRIPCIÓN DETALLADA DE DIFERENTES TIPOS DE PRUEBAS.

Avaible at: https://es.wikipedia.org/wiki/Prueba_unitaria
https://keepcoding.io/blog/que-son-las-pruebas-de-integracion/#Que_son_las_pruebas_de_integracion

https://www.zaptest.com/es/que-son-las-pruebas-funcionales-tipos-ejemplos-lista-de-comprobacion-y-aplicacion

https://es.parasoft.com/blog/continuous-integration-with-parasoft-soatest/

https://www.encora.com/es/blog/pruebas-de-rendimiento-cuando-y-como

TECNICAS DE TESTING

Avaible at: https://www.paradigmadigital.com/dev/tdd-como-metodologia-de-diseno-de-software/

https://apiumhub.com/es/tech-blog-barcelona/beneficios-de-tdd/

https://www.hiberus.com/crecemos-contigo/bdd-behavior-driven-developement/

https://apiumhub.com/es/tech-blog-barcelona/desarrollo-guiado-por-comportamiento/

AUTOMATIZACIÓN DE PRUEBAS

Avaible at: https://ilimit.com/blog/beneficioss-automatizacion-pruebas/

https://ciberninjas.com/10-mejores-herramientas-pruebas-ui/

https://www.adictosaltrabajo.com/2016/03/30/introduccion-a-protractor/

https://www.apium.io

https://es.wikipedia.org/wiki/JUnit