Mario Sánchez

Informe testing

¿Que es el testing?

El testing se utiliza para revisar los códigos y asegurarse de que no haya errores, garantizando así la eficacia del software en un tiempo relativamente reducido a que si no se utilizase, o incluso que se puedan evitar dichos errores.Por lo que esto se ha vuelto un oficio muy necesario en el mundo de la programación. A las personas que se dedican a este oficio de probar programas y códigos se les llama testers.

Las pruebas de código y el testing no son del todo lo mismo, existen diferencias:

- Testing: Método para verificar que un producto digital cumple con los requisitos esperados
- Pruebas de código: Proceso de validación del funcionamiento del software de un determinado programa o app.

Testing	Pruebas de código
● Tipo de prueba no funcional ● Determina la velocidad, estabilidad y escalabilidad de una app de software ● Verifica el rendimiento de una aplicación ● Ejm: Pruebas de carga y de esfuerzo	● Conocido como pruebas de software ● Pruebas manuales ● Comprueban que un fragmento de código está funcionando ● Aislan una parte de código ● Suelen ser la primera evaluación de un código

Al realizar pruebas de código , nos aseguramos de que el código funcione correctamente hasta donde estamos, o por partes. Nos permite alcanzar las expectativas que se esperan de nuestro código además de acelerar el proceso de creación del mismo.

Tipos de pruebas de código

Existen diferentes tipos de pruebas de código con las cuales se puede trabajar de diferentes maneras:

- **Pruebas unitarias**: Pruebas de bajo nivel , realizadas cerca de la fuente de la aplicación. Prueba métodos y funciones individuales de las clases o módulos.Algunas herramientas usadas son:Cactus,Mockito, Jetty, Dumster,...
- **Pruebas de integración**: Verifican que los distintos módulos o servicios utilizados en tu aplicación funcionan bien juntos.Algunas herramientas para esta prueba son: Selenium,Postman,Watir, Jest,...
- **Pruebas de sistema**: Se encargan de validar el sistema en base a ciertas especificaciones.Algunas herramientas son:

- **Pruebas de aceptación**: Pruebas que comprueban si un sistema alcanza los requisitos que se le piden por parte de las empresas. Replican las conductas de los usuarios para comprobar el funcionamiento del sistema.Algunas herramientas son:Cucumber, Fitnesse,...
- **Pruebas de estrés**:Esta prueba se centra en determinar cuales son los límites del programa o del sistema, encontrando así su punto de inflexión. Algunas herramientas son:Atentus, Neoload,...
- **Pruebas de extremo a extremo**: Replica el comportamiento de un usuario en el software/ app completo. También verifica el funcionamiento de la app según lo previsto desde lo más sencillo hasta lo más complejo.Algunas herramientas son:Selenium, Cypress,Playwright,...
- **Pruebas de rendimiento**: Evalúa la funcionalidad de un sistema con cierto estrés , no el máximo que se pueda usar, sino con una cantidad de uso específica. Con esto se puede medir su velocidad , fiabilidad, la capacidad de respuesta,...Esto lo hace midiendo al mismo tiempo varias solicitudes o con cierta cantidad de datos.Algunas herramientas son:WebLoad, Stormforge,Eggplant,...
- **Pruebas de humo**: Sirven para comprobar el funcionamiento básico del programa, ejecutándose así rápidamente. Así se comprueba la seguridad de que te proporcione datos, te asegura al menos el funcionamiento básico.Algunas herramientas son: Jenkins

Técnicas de testing

- **TDD** (Test Driven Development:Dirige el desarrollo de un producto por medio de pruebas. Con esto se consigue un código más tolerante al cambio, robusto y seguro. Esto se usa cuando los requisitos de un producto no están definidos y cabe la posibilidad de hacer cambios.
- **BDD**(Behavior Driven Development): Surge del TDD.Hace posible la colaboración de equipos técnicos o testers.
- **IC**(Integración continua):Los cambios que se realizan en el código, se guardan y se actualizan varias veces al día. De esta manera, se comparten cambios en todos los dispositivos que estén trabajando con ese código.
- **DC**(Despliegue continuo): Esta técnica se centra en ir guardando todo el rato los cambios que se realizan en el código, cambiando la página web a la par.
- **TAU**(Test de aceptación de usuario):Este método es una serie de pruebas en las cuales se aseguran que el código y sus funciones están a disposición del usuario.

Automatización de pruebas

Las pruebas automatizadas son el uso de herramientas de software para automatizar el proceso manual de revisión y validación de un software/programa/app. La automatización de las pruebas nos sirve para , por ejemplo, ir guardando constantemente un programa y que a la vez se actualice la página al mismo tiempo.

Herramientas comunes

- **Selenium**: Usado para probar un mismo código en diferentes navegadores y plataformas, siendo también una herramienta muy flexible entre diferentes lenguajes de programación.
- **Cypress:** Herramienta para la automatización de las pruebas de las páginas web.(TDD)
- **Appium:** Puede realizar diferentes pruebas complejas, compatible con varios lenguajes y dispositivos.
- **Playwright:** Automatiza pruebas de aplicaciones web, funcionando solo con navegadores principales. Es una herramienta rápida y confiable
- **Serenity:** Framework basado en java.Facilita la escritura de pruebas de aceptación y regresión. También administra el estado entre pasos, hace capturas de pantalla,ejecuta pruebas al mismo tiempo que se trabaja,...
- **Robot framework:** Puede probar sistemas de software de cualquier tipo. Este tipo de automatización cuenta con la ventaja de que se puede adaptar a las necesidades de cualquier equipo de desarrollo.

Ejemplos

- **Pruebas unitarias**: Desarrollo de una función de cálculo de descuentos en un sistema de comercio. En este caso, se escriben pruebas unitarias con el fin de que los cálculos se calculen y se distribuyan entre diversos sistemas.
- **Pruebas de integración**:Integración de un sistema de pago con un sistema de gestión de pedidos en una aplicación de comercio. Esto se hace para verificar que los datos pueden llegar de un sistema a otro sin complicaciones y que la gestión funciona correctamente.
- **Pruebas de aceptación del usuario**: Implementación de un sistema de reservas para una compañía de viajes. Se prueba el sistema para comprobar que sea sencillo de usar para reservar.

Conclusión

El testing es importante para revisar que el trabajo o los proyectos se hacen correctamente y atendiendo a las expectativas que se tienen y se pueden llegar a tener. También cabe resaltar que es muy accesible lo del testing debido a la gran cantidad de opciones de uso que contiene.También, cabe destacar que las pruebas de código son muy importantes para no arrastrar fallos que se han cometido en el pasado.Por otro lado, la automatización de pruebas es importante ya que nos puede

ayudar a ahorrar tiempo y recursos. Así podemos trabajar a la par que la página se va revisando y actualizando ayudando a que todo funcione correctamente.

Fuentes

https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/

https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing

https://keepcoding.io/blog/que-son-las-pruebas-unitarias-de-software/

https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing

https://atentus.com/2022/11/08/pruebas-de-estres/#:~:text=%C2%BFQu%C3%A9%20es%20una%20prueba%20de,aplicaci%C3%B3n%20como%20de%20su%20infraestructura.

https://unaqaenapuros.wordpress.com/2021/04/28/072-pruebas-de-integracion-iii-herramientas/

https://www.hiberus.com/crecemos-contigo/las-mejores-herramientas-para-cada-tipo-de-testing/

https://softwarecrafters.io/javascript/tdd-test-driven-development

https://spain.girlsintech.org/que-tecnicas-de-testing-software-debemos-usar/

https://openexpoeurope.com/es/automatizacion-de-frameworks-de-codigo-abierto/