DP II - 2021-2022

LESSONS LEARNT ABOUT TESTING

https://github.com/ddiazlop/Acme_Recipes_G10

Miembros:

- Carolina Carrasco Díaz
- Diego Jesús Díaz López
- Julio Navarro Rodríguez
- Salvador Parejo Ramos
- María Isabel Pedraz Rodríguez
- Alvaro Sevilla Cabrera

Tutor: RAFAEL CORCHUELO GIL

GRUPO - G10

1 Índice

2 Resumen Ejecutivo

Para la realización de este documento se han tenido en cuenta las herramientas disponibles en la enseñanza virtual y las distintas fuentes visitadas, además de la experiencia obtenida a lo largo de la ejecución del proyecto y la realización de las prácticas.

3 Tabla de Versionado

Versión	Fecha	Descripción
1.0	04/09/2022	Realización del documento
1.1	06/09/2022	Modificados algunos puntos para un mejor entendimiento de ellos y añadidas lecciones aprendidas
1.2	07/08/2022	Añadidas bibliografía y conclusión
2.0	08/08/2022	Finalización, revisión y entrega del documento.

4 Introducción

A continuación en este documento nos centraremos en las lecciones aprendidas sobre el testeo de la aplicación. así como inicialmente recordar su significado y su vital importancia en el proyecto.

Recordamos que el testing es el proceso mediante el cual se evalúa y se verifica que una aplicación y todos sus componentes realizan correctamente la función para la que han sido creados.

Esto aporta grandes beneficios al proceso software como la prevención de errores, mejora del rendimiento y reducción de los costes de producción y mantenimiento.

5 Concepto SUT

En primer lugar se definirá el concepto de SUT (System Under Test): lo cual podemos definir como "todo aquello que se somete a la prueba".

Por cada test hay un único sistema sobre el que se hace la prueba, es decir, un único SUT, de manera que se pueda obtener un resultado sobre el funcionamiento del test, ya sea positivo o negativo. De esta forma, se puede observar si el resultado obtenido es el esperado o no.

Estas pruebas de test se realizan bajo unas condiciones definidas previamente y se le aplica una serie de estímulos con la finalidad de detectar si el nivel de calidad es adecuado o no.

6 Funcionamiento de la herramienta: Testing

Las herramientas de testing proporcionan una solución a la hora de dinamizar las pruebas que se realizan a los sistemas. *[3][4]*

A la hora de realizar los test, se tiene en cuenta la importancia que estos tienen a la hora de desarrollar un sistema software, como por ejemplo:

- Establecer la calidad del producto y poder actuar sobre los fallos detectados, lo que permite un ahorro a posteriori en costes y en tiempo.

- Comprueba los valores solicitados, favoreciendo así la obtención de un resultado acorde con lo esperado.

- Reduce costes de mantenimiento, ya que realizando previamente los test se evalúa la herramienta antes de su puesta en marcha.

- Permite identificar las causas de los errores, de manera que a posteriori sea mucho más sencillo su resolución, en caso de que se volvieran a identificar.

➔ NEGATIVE TESTS CASES AND POSITIVE TESTS CASES

A la hora de realizar los test, se han realizado tests negativos y positivos, de manera que se puedan analizar las posibilidades que puedan influir en los resultados de los test, tanto positivas como negativas. *[5]*

Los casos de prueba positivos garantizan que los usuarios puedan realizar las acciones adecuadas al utilizar datos válidos. Los casos de prueba negativos se realizan para intentar "romper" el software realizando acciones no válidas (o inaceptables), o utilizando datos no válidos.

Algunos ejemplos de casos positivos:

- Se puede crear un nuevo plato (fine-dish)
- Se puede crear un nuevo utensilio de cocina (kitchenware)

Algunos ejemplos de casos negativos:

- Se intenta crear un nuevo memorando (memoranda) sin indicar el plato (fine-dish)
- Se intenta eliminar un memorando (memoranda) creado.

Por tanto, podemos finalizar sabiendo que los casos positivos son aquellos que incluyen acciones que se pueden realizar y datos válidos y los casos negativos son aquellos que incluyen acciones que no se pueden realizar y datos no válidos.

7 Diferentes tipos de pruebas software sobre una aplicación

A la hora de realizar un estudio de la aplicación, se han categorizado los tests, facilitando así, además, su análisis, siguiendo la siguiente clasificación.

- Test de aceptación: Su función principal es comprobar que un software realiza su función correctamente.
- Test de integración: Su función es comprobar que todos los componentes de un software funcionan bien cuando se encuentran juntos.
- Tests Unitarios: Su función es comprobar que todos los componentes de un software realizan su función correctamente.
- Test de Funcionalidad: Comprueba que todas las funcionalidades funcionan correctamente simulando escenas de negocio basadas en los requisitos funcionales.
- Test de Rendimiento: Comprueba cómo funciona el producto software frente a diferentes cargas de uso.
- Test de Regresión: Comprueba cómo afectaría negativamente el añadir nuevas funcionalidades al rendimiento o al funcionamiento de las funcionalidades ya integradas.
- Test de estrés: Comprueba cuánta carga puede soportar el sistema software hasta fallar.
- Test de Usabilidad: Comprueba si un usuario es capaz de utilizar una funcionalidad en la web.

➔ End-To-End testing

Podemos definir el siguiente concepto: *End to end Testing* (Test de extremo a extremo) como una metodología que permite evaluar el funcionamiento y probarlo desde su comienzo hasta su final. Verifica que todos los componentes del sistema corren y se llevan a cabo de forma óptima bajo unos escenarios que sean reales. De esta forma se puede simular una situación a la que se vea sometida el sistema de forma automática. *[1][2]*

Además, ha permitido realizar también una tabla comparativa sobre la velocidad a la que se procesan los test en un equipo y en otro, permitiendo realizar una comparación entre cómo se ejecutan los tests en ambos. De esta forma, se obtienen unos resultados que, en función del equipo sobre el que se está realizando el testing, pueden ser mejores o peores.

8 Performance Testing

En cuanto a los tests de rendimiento (performance testing), nos centramos en el tiempo que tarda el software en gestionar una petición individual, no un proceso completo ya que eso depende del número total de procesos individuales que haya que ejecutar.

Los *benchmarks* consisten en medir el tiempo que se tarda en resolver una petición desde que se realiza, se procesa y se devuelven los resultados. Podemos definir este concepto como pruebas de velocidad, resistencia, inteligencia y aguante para procesadores y memorias.

Para mejorar los resultados de estos test se lleva a cabo refactorización que comprende el hacer cambios en el proyecto para que los métodos mantengan la funcionalidad pero de forma más eficiente.

9 Importancia de la calidad y cobertura de los tests

La calidad y la cobertura de los test identificadas mediante las diferentes aplicaciones, permiten establecer el alcance de las pruebas, ya que se deben considerar las posibles opciones para establecer una cobertura óptima y lo más amplia posible para que los resultados generados por los test sean veraces, asegurando que sea una solución sin fisuras y que generen una idea real del estado de la aplicación. [6][7]

Con este tipo de herramientas se puede indicar si se ha realizado correctamente una prueba o no, en función de un criterio establecido. Gracias a esto se puede:

- Medir la calidad de los tests
- Determinar el momento de finalización
- Ver lo que se ha probado y analizar lo que queda por probar

A mayor cobertura mayor será el examen que se haya realizado al sistema, por lo que se podrá detectar de una forma mejor el alcance del testeo realizado.

Por tanto, una buena calidad y cobertura de los test permite:

- Identificar errores en etapas tempranas
- En caso de que se detecte que la cobertura no es apropiada, se podrían generar más casos de prueba para mejorar la calidad y la cobertura

- Eliminar casos de prueba irrelevantes o innecesarios, de manera que el código generado sea más ligero

- Eficiencia, ya que permite ahorrar tiempo y costes con una buena cobertura de los test.

- Pocos errores a posteriori, lo que permite un ahorro en tiempo y dinero.

10 Lecciones aprendidas

Finalmente, hacemos alusión a que todos los miembros hemos recordado la importancia del testeo de la aplicación, así como recordar su vital importancia dentro del proyecto, adquiriendo un enfoque de cómo testear un proyecto de diferentes formas.

A la hora de llevarlo a la práctica, todos los miembros del equipo hemos realizado todos tipos de tests, ya que cada uno ha realizado tests de las funcionalidades que ha implementado y/o todas las que hicieran falta implementar para garantizar la cobertura del código del proyecto.

Gracias a esta funcionalidad requerida, se ha puesto en valor la necesidad de realizar los tests a las aplicaciones y sistemas implementados, generando una necesidad clara y evidente de realizarlo en un futuro en el desarrollo de los proyectos en los que se esté involucrado.

11 Conclusión

Concluimos con que se han cumplido y superado los conocimientos de la asignatura, así como el objetivo de las tareas a la hora de la implementación de los tests y, como se ha mencionado anteriormente, valorar el uso de estas herramientas a lo largo de la ejecución de los proyectos.

Gracias a estas herramientas se pueden realizar pruebas de software adecuadas, garantizando una cobertura óptima y con una calidad adecuada al proyecto en desarrollo.

12 BIBLIOGRAFÍA

[1]: Gillis, A. S. (2018, 28 febrero). End-to-end testing. SearchSoftwareQuality. https://www.techtarget.com/searchsoftwarequality/definition/End-to-end-testing

[2]: Perforce (2020, 13 Agosto), Attention Required! | Cloudflare. (s.f.). https://www.perfecto.io/blog/comprehensive-guide-end-end-e2e-testing

[3]: Perforce (2018, 04 Julio) SUT, fake, stub, mock y spy en las pruebas con un ejemplo. https://picodotdev.github.io/blog-bitix/2018/07/que-es-un-sut-fake-stub-mock-y-spy-en-las-pruebas-con-un-ejemplo/

[4]: SG Buzz. Fundamentos de la Prueba de Software: conceptos, justificación y alcance. https://sg.com.mx/revista/4/fundamentos-prueba-software-conceptos-justificacion-y-alcance

[5]: Kathy Claycomb (2018, 09 Abril) Positive vs Negative Test Cases. https://netmind.net/en/positive-vs-negative-test-cases/#:~:text=Positive%20test%20cases%20ensure%20that,or%20by%20using%20invalid%20data.

[6] Toledo, F. (2018, 28 julio). Cobertura de pruebas. https://www.federico-toledo.com/que-es-la-cobertura-de-pruebas/

[7] Andrade, A. (2021, 3 septiembre). La cobertura de pruebas unitarias. https://alexandrade.net/blog-de-ingenieria-de-software/calidad-de-software/por-que-la-cobertura-de-pruebas-unitarias-es-una-parte-importante-de-qa/#Que_es_la_cobertura_de_pruebas_unitarias