TESTING REPORT

Repositorio:
https://github.com/marsamber/Acme-One

Miembros (Grupo E3.05):
Gonzalo Martínez Martínez (gonmarmar5@alum.us.es)
Álvaro Miranda Pozo (alvmirpoz@alum.us.es)
Pedro Parrilla Bascón (pedparbas@alum.us.es)
Marta Sampedro Bernal (marsamber@alum.us.es)
Álvaro Vázquez Ortiz (alvvazort@alum.us.es)
María Castro Bonilla (marcasbon@alum.us.es)

2 de junio de 2022

Índice

Resumen Ejecutivo

Este documento recopila, resume y analiza las diferentes competencias adquiridas en el ámbito de “testing” durante el desarrollo tanto del proyecto como del curso académico en lo que a esta asignatura (Diseño y Pruebas II) respecta. De esta forma obtenemos un documento que recoge los conocimientos aprendidos e interiorizados por el equipo, obteniendo una especie de “feedback” que nos ayude a comprender la utilidad que provee el “testing”.

Historial de versiones

Versión	Fecha	Descripción
1.0	29-05-2022	Creación del documento

Introducción

En este documento se hablará de lo que hemos aprendido de testing gracias a la asignatura y al proyecto Acme Toolkits.
Primero sería importante saber diferenciar entre un bug y un fallo. Un bug es algo que está mal en el código, el objetivo de los tests funcionales es el de encontrar todos los bugs posibles. Un fallo, sin embargo, es el efecto de un bug en tiempo de ejecución, los tests funcionales detectan los fallos comparando los resultados reales con los esperados.
Otros conceptos relevantes son el de depuración (se rastrea un fallo hasta los errores que lo han causado), errores (mensaje mediante el cual un sistema informa de que algo no se puede hacer debido a problemas en la solicitud), pánicos (se produce cuando se lanza una excepción) y peticiones legales e ilegales (solicitud que no puede realizarse de acuerdo con los requisitos).
En un banco de tests podemos encontrar casos positivos y casos negativos.
Un caso positivo es un script que comprueba que el sistema funciona bien en un contexto en el que se espera que funcione bien. Funcionar bien significa que el sistema no emite ningún error o excepción y produce los resultados esperados.
Un caso negativo es un script que verifica que el sistema funciona bien en un contexto en el que se espera que informe de un error.
Los tests se pueden clasificar en diversos tipos según el momento, objetivo, opacidad, etc.
En este informe se tratarán los diferentes tipos de testing que hemos efectuado durante el transcurso de la asignatura: funcionales y de rendimiento.

Contenido

Podemos dividir en primer lugar los tipos de pruebas en 2 grandes grupos:

Tests funcionales

Son las pruebas que analizan si el código se ejecuta de manera adecuada, produciendo salidas esperadas o predecibles en todo momento. Otra forma de describirlos sería como los requisitos funcionales definidos por el cliente, los cuáles son especificados explícitamente y definen como tal el funcionamiento del software/producto desarrollado.

En nuestro caso, se realiza testeo E2E (end-to-end, de extremo a extremo), el objetivo es simular la interacción de un usuario con la interfaz web mientras se verifica que el sistema se comporta como es esperado. Si no fuera así, habríamos encontrado un fallo que implicaría tener que depurar el código para encontrar los correspondientes bugs y solucionarlos. Esta nueva solución deberá someterse a las pruebas de nuevo para corroborar su validez.

Siempre es importante intentar cubrir el máximo de código posible al testear, de este modo nos podremos asegurar con mayor certeza de que el sistema funciona como es debido. Esta cobertura podremos visualizarla en nuestro proyecto gracias a la herramienta “CoverageRunner”, un plug-in de Eclipse.
Por otra parte, también es conveniente cubrir todos los datos factibles: poblando con datos de ejemplo, listándolos y en sus formularios.

Todos nuestros tests heredan de la clase AbstractTest indirectamente. Por defecto, se eliminan todas las cookies y se navega hacia la página principal antes de comenzar el test. Esta clase tiene una gran variedad de métodos mediante los cuales podemos realizar multitud de acciones simulando a un usuario navegando por nuestra web.

La clase TestHarness extiende de AbstractTest, y es de esta que extienden nuestros tests. En ella tenemos los métodos útiles para este proyecto, como el inicio de sesión, el registro y el cierre de sesión.

En todos nuestros tests listamos una entidad, comprobamos una a una que sus datos corresponden con los esperados, y hacemos lo mismo con sus detalles. De esta forma estamos revisando tanto las listas como los formularios poblados con los datos de ejemplo.
En el caso específico de un test de creación, se crea nuestra nueva instancia de la entidad, y luego verificamos que aparece en la lista y procedemos a efectuar lo anteriormente mencionado; en los tests de actualización, se actualiza la instancia y se hacen los mismos pasos que en el de creación; finalmente en los de publicación, se verifica que no origine ningún error la acción de publicar.

Por el contrario, en los tests negativos, comprobamos que estos objetos no se crean o actualizan en el caso de meter datos que no se permiten según el modelo definido.

Tests de rendimiento

Estas pruebas analizan el código desde el punto de vista de la productividad, podríamos decir. Otra manera de definirlos podría ser como los requisitos no funcionales descritos por el cliente. Estos suelen definirse como atributos de calidad en la mayoría de los casos, como por ejemplo el tiempo de respuesta esperado del sistema o funcionalidad concreta.

Conceptos importantes:

- “Benchmarking” o evaluación comparativa.
  Este proceso consiste en la medición del tiempo que se toma para responder a una petición. Incluyendo el envío de la misma, procesamiento y entrega de los resultados.

- En este caso, un fallo es una desviación en el tiempo de salida esperado.
- El análisis de rendimiento de un proyecto o “profiling” es la medición de los tiempos en los que nuestro proyecto invoca a los métodos que contiene durante una prueba de rendimiento.
- El análisis de rendimiento de un ordenador requiere de la medición de la utilización de los principales componentes durante un test de rendimiento. Esto incluye tanto recursos de cpu, almacenamiento, Ram, caché, etc…
- La refactorización u optimización es cambiar el código del proyecto de forma que, realizando la misma funcionalidad, se haga más rápido o eficientemente.
- El Análisis estadístico busca determinar si las conclusiones de un ejemplo de prueba, se puede extrapolar a su población con cierto grado de certeza.

Todos los resultados de estos tests deben quedar reflejados en los archivos conocidos como “logs” o registros. A través de estos luego podremos realizar un estudio estadístico centrado en la característica que más nos interese (principalmente el tiempo de respuesta o “wall time”).

Por último el análisis estadístico conlleva un proceso complejo que requiere de la ejecución de múltiples pruebas, colección de datos y corroboración de hipótesis, las cuáles se hacen para predecir los resultados del testing hecho sobre el código analizado y ahora refactorizado. De este modo obtenemos unas medidas con cierto grado de confianza que nos sirven para proporcionar un mejor análisis de nuestro proyecto.

Conclusión

En esta asignatura hemos aprendido a realizar distintos tipos de tests que nos ayudan a cubrir gran parte de nuestro código, ya que ahora podemos hacer tests para todas las funcionalidades principales de nuestra web, de forma que todas nuestras operaciones CRUD pueden probarse y verificar su correcto funcionamiento. Por último hemos tenido un primer contacto con las herramientas de testing tanto funcional como de rendimiento lo que nos ha proporcionado una mejor visión del tratamiento de las pruebas y su importancia.

Bibliografía

Transparencias de la asignatura