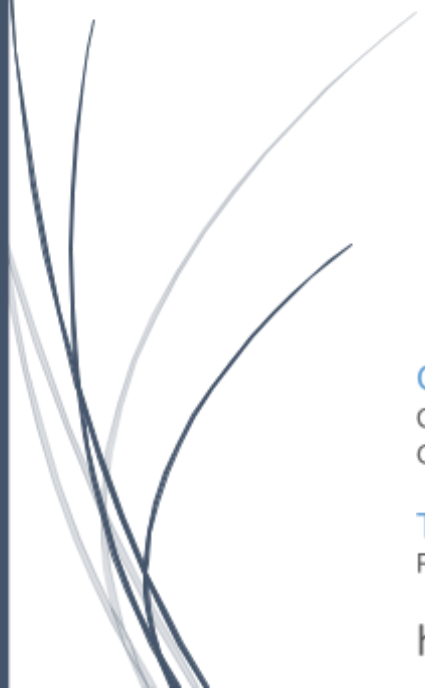
Report About Testing

21-11-2022

Noviembre D05 – Diseño y Pruebas II



Grupo D07

CERRATO SÁNCHEZ, LUIS (luicersan@alum.us.es)

GUITÉRREZ CONTRERAS, ERNESTO (erngutcon@alum.us.es)

Tutor

PATRICIA JIMÉNEZ AGUIRRE

https://github.com/erngutcon/Acme-Courses-D05

Índice

Resumen

En este documento se muestran los conocimientos que hemos obtenido sobre los test durante la realización de los entregables de la asignatura Diseño y Pruebas II.

A la hora de realizar un proyecto software es necesario realizar pruebas para poder comprobar que los métodos implementados y las funcionalidades funcionan correctamente devolviendo los resultados esperados.

Es necesario tener un conjunto de pruebas ya que al ejecutarlas el sistema prueba distintos casos de prueba y nos garantiza que todo funciona según lo previsto.

De esta forma, si los test fallan nos están indicando que hay errores por corregir, mientras qué si todos los test salen correctos, podemos estar tranquilos.

Tabla de revisión

V1.0	21-11-2022	Informe sobre las pruebas D05

Introducción

Tal y como se describe en el resumen, este documento se centra en los conocimientos obtenidos sobre las pruebas tras haber realizado los anteriores entregables de la asignatura Diseño y Pruebas II.

Las pruebas las hemos realizado para comprobar que los métodos funcionan según lo esperado. Cada prueba está asignada a una funcionalidad. Y mediante ellas hemos podido encontrar algunos fallos para luego poder corregirlos y así funcione todo de manera correcta.

En una clase test se pueden implementar:

- Positive Test: son aquellas pruebas que se encargan de comprobar que un caso en concreto funcione bien. Que no existen errores ni excepciones y devuelve el resultado esperado.
- Negative Test: son aquellas pruebas que hacen que cuando algo no va bien se comprueba que se devuelva un informe de error.

Está bien implementar ambos tipos de prueba ya que así se comprueba tanto que se devuelva el resultado esperando como que se informe del error en caso contrario.

Contenido

Tipos de pruebas:

- Según el momento en el que se realizan:
    - Estáticas
    - Dinámicas
- Según el objetivo de prueba:
    - Los requisitos funcionales, se comprueban los resultados esperados
    - Los requisitos no funcionales, se comprueba el rendimiento
- Según la opacidad:
    - White-box: se conocen las partes internas del sistema
    - Black-box: solo se conoce la interfaz
- Según la motivación:
    - Encontrar fallos en el código
    - Lograr que el cliente esté conforme
- Según la cobertura:
    - Cubrir tantos datos, mientras mayor sea la cobertura de los datos más se podrá confiar en el sistema. Primero se ejecuta el populate sample para llenar la base de datos, luego el listado de datos y los datos de formulario.
    - Cubrir tantas rutas de ejecución, mientras más rutas se cubran más se puede confiar en el sistema. Primero se exploran las secuencias, luego los condicionales y por último los bucles.
- Según los supuestos
    - Pruebas unitarias
    - Pruebas de integración

Las pruebas que realizamos son de extremo a extremo ya que se realiza una simulación como si el usuario estuviera interactuando con el sistema, de esta forma se comprueba que todo funciona según lo esperado. Esto se realiza usando el controlador Gecko.

En el caso de las pruebas estáticas no usamos E2E, para detectar bad smells usamos un plugin de eclipse llamado SonarLint.

Para las pruebas de black-box no necesitamos realizar pruebas ya que tenemos acceso al código fuente.

Performance testing

Este tipo de pruebas se centra en comprobar los requisitos no funcionales, el rendimiento del sistema.

Una vez creado los casos de prueba positivos y los casos de prueba negativos. Se generan registros de rendimiento, estos quedan guardados en una carpeta llamada logs. Estos se pueden consultar desde eclipse o bien desde Excel.

Hemos visto los request logs, que aportan información sobre las peticiones que se realizaron (el momento de la petición, la ruta y la duración en milisegundos). Y los test-cases logs, estos muestran información sobre la ejecución de los casos de pruebas (el momento, la clase del test, el caso positivo y el caso negativo, la duración, una descripción y el resultado).

A partir de los datos obtenidos se pueden hacer análisis de rendimiento y estadísticas.

Conclusiones

Para comprobar que un proyecto software funciona correctamente, no tiene fallos ni errores y devuelve los resultados que se esperan es necesario la creación de los tests. Así nos aseguramos de que todo funciona según lo previsto y no se darán fallos ni excepciones.

Es importante que se comprueben tanto los casos positivos como los negativos. Así tendremos seguridad de que se devuelven los resultados esperados sin fallos ni errores, así como comprobar que la aplicación muestra los errores cuando se espera que los muestre.

Otro aspecto importante son los test de hacking, los cuales comprueban que nuestra aplicación sea segura y que los usuarios no puedan acceder a partes de esta a las que no tienen los permisos necesarios.

Con la simulación de los casos de prueba nos aseguramos qué todo funciona según lo previsto.

El rendimiento del sistema lo analizamos mediante el performance report, haciendo pruebas en nuestros dos equipos personales, teniendo así una mejor idea del rendimiento de la aplicación.

Bibliografía

Diapositivas de la asignatura tomadas de enseñanza virtual.