ACME-TOOLKITS

GROUP WIS ARCHITECTURE TESTING KNOWLEDGE

ACME - TOOLKITS

Repositorio: https://github.com/danrodcam/Acme-Toolkits.git
Fecha: 01/06/2022

Miembros del grupo:

- Fernando Andres Galindo
- Francisco Murillo Prior
- Daniel Rodríguez Camacho
- Alvaro Rodríguez García
- Antonio Rosado Barrera
- Jose Manuel Ruiz Perez

ÍNDICE

RESUMEN EJECUTIVO

En este documento redactamos los conocimientos que tiene el grupo sobre el testing de la arquitectura WIS obtenidos en las asignaturas previas que hemos cursado. Para ello hemos realizado una reunión donde cada uno ha expuesto sus ideas y aquí es donde se recogen y unifican.

En este documento redactamos los conocimientos aprendidos a lo largo de la asignatura de Diseños y Pruebas II por parte del grupo sobre el testing de la arquitectura WIS. Para ello hemos realizado una reunión donde cada miembro del grupo ha expuesto sobre sus ideas aprendidas a lo largo de la asignatura, se han recopilado todas las ideas y en este documento se han unificado.

Control de versión del documento:

Versión del documento	Fecha	Descripción
1.0	01/06/2022	Creación y finalización del documento

1. Introducción

El testing en el software, es un proceso de verificación y validación sobre la funcionalidad de un programa o una aplicación de software con el objetivo de garantizar que el producto de software esté libre de defectos. Dicho esto, este documento recoge todo lo aprendido por los miembros del grupo. En esta asignatura hemos aprendido sobre testing funcionales, testing performance y sobre SonarLint.

2. Conocimientos de testing

- Los conocimientos sobre el testing funcional sobre la arquitectura WIS que hemos ido adquiriendo como grupo se van a dividir en 2 partes:

- **Testeo Informal:** Antes de escribir sus pruebas formales, es una buena idea realizar algunas pruebas informales, que definitivamente nos ayudarán a desarrollar más eficientemente y con un mayor grado de comprensión los próximos test.

  Los test informales se han realizado teniendo como referencia los csv del proyecto y comprobando mediante ello que las funcionalidades creadas estaban mostrando los datos correctamente.

- **Testeo Formal:** Los casos de prueba formales consisten en scripts que utilizan el controlador Gecko para simular que un usuario interactúa con Firefox (marionette mode) y leyendo los resultados devueltos por su aplicación para verificar que sean los esperados.

  En las pruebas formales se crean distintos entornos de prueba, donde crearemos varios csv. Podremos catalogar los distintos entornos de prueba en casos positivos y casos negativos.

  Para los casos positivos escogeremos algunos elementos de nuestro sample data que serán los elementos de prueba. Estos elementos deben cumplir todas las validaciones compuestas tanto por las anotaciones de Spring-Boot tanto como por las validaciones más complejas implementadas por nosotros. Esto es debido a que los test positivos no deben devolver errores.
  Para los casos negativos añadiremos elementos dentro de un csv que no cumplan con todos las validaciones, para que de esta forma se devuelva algún error en específico.

- Los conocimientos sobre el testing de rendimiento sobre la arquitectura WIS que hemos ido adquiriendo ha sido la comparativa a realizar entre 2 ordenadores de 2 miembros distintos del grupo sobre el reporte realizado sobre el rendimiento de los test realizados. De esto hemos aprendido que es para obtener medidas cuantitativas sobre el rendimiento del proyecto así como un porcentaje de desviación entre el rendimiento de ambos equipos.

- Por último, los conocimientos obtenidos sobre el testeo de bugs y bad smells en el código se basan en la nueva herramienta que hemos utilizado para la detección de estos bugs denominada SonarLint. SonarLint nos muestra todos los tipos de errores relacionados con nuestro proyecto, y además nos indica y nos desplaza al error dentro del proyecto para solucionar dicho error, por lo que es bastante útil.

3. Conclusiones

Como conclusión, todos los miembros del grupo hemos aprendido y profundizado sobre los distintos escenarios de prueba dentro de los tests funcionales. Todos hemos participado dentro del desarrollo de los distintos tests funcionales y por lo tanto no tener que depender tanto de otros miembros a la hora de elaborar tests. Además, se ha aprendido a realizar un reporte de rendimiento sobre cualquier proyecto, así como a realizar una comparativa del rendimiento

con otro reporte de rendimiento realizado en un ordenador diferente. Por último, hemos aprendido a detectar errores de código mediante SonarLint que nos va a ser útil en un futuro.

4. Bibliografía

Intencionadamente en blanco