TESTING REPORT

Grupo E4.01

https://github.com/DP2-E4-01/D01-Acme-Toolkit

02/03/2022

Integrantes:

Daniel Díaz Nogales	(dandianog@alum.us.es)
Luis Miguel Bellido Zancarrón	(luibelzan@alum.us.es)
Diego González Quintanilla	(diegonqui@alum.us.es)
Eloy Moreno Dominguez	(elomordom@alum.us.es)
José Mª García Quijada	(josgarqui@alum.us.es)
Juan Antonio Mena Vargas	(juanmenvar@alum.us.es)

Versión	Descripción	Fecha
v1.0	Creación inicial	28/02/2022
v2.0	Revisión final del informe	02/03/2022

Resumen ejecutivo

Debido a la necesidad de crear un documento donde dar a conocer los conocimientos del grupo sobre el testing, se desarrolla el presente documento.

Con el objetivo de resolver dicho problema, el documento se divide en una sola sección donde los integrantes del grupo aportan las ideas que tienen acerca del testing y las ponen en común.

Para finalizar podemos afirmar que, tras crear este informe, hemos adquirido nuevos conocimientos sobre el testing, los tipos de pruebas que se pueden crear y la cobertura de pruebas.

Introducción

Gracias al conocimiento adquirido en otras asignaturas, podemos poner en común las distintas ideas que tenemos los integrantes de este documento sobre la realización de pruebas. Principalmente hemos desarrollado, sobre todo en el entorno Eclipse, las pruebas de código gracias a la herramienta JUnit, integrada en el propio programa. Con ella podíamos ejecutar pruebas a fragmentos de código previamente realizados.

Conocimiento de los integrantes acerca del testing

Para empezar, podemos diferenciar entre dos tipos de pruebas: **funcionales** y **no funcionales**. En las pruebas **funcionales**, se ejecutan tests para probar el correcto funcionamiento del código que satisfagan los requisitos e historias de usuario, mientras que las pruebas **no funcionales** se centran en garantizar la calidad del software, probando aspectos como la seguridad, fiabilidad, usabilidad, escalabilidad, mantenibilidad…

Dentro de las pruebas funcionales podemos distinguir entre:

- **Pruebas unitarias:** Tests que se realizan a métodos o funciones concretas dentro de las distintas clases que se crean en un proyecto.
- **Pruebas de integración:** Estas pruebas implican probar los diferentes módulos de una aplicación y las conexiones que existen entre ellos. Ej: Probar que una aplicación está enlazada a una base de datos.
- **Pruebas de vista:** Garantiza que las implementaciones funcionales de la aplicación se estén ejecutando correctamente en la interfaz mostrada a un navegador web.
- **Pruebas de carga**: Se prueba la aplicación con el número de conexiones esperadas para ver si las soporta, o se busca conocer los límites de conexiones que permite la configuración del servidor y la aplicación.

Las pruebas con las que estamos más familiarizados y de las que tenemos más conocimiento son las **unitarias**, ya que hemos desarrollado dichas pruebas en diferentes asignaturas, como por ejemplo en DP1, donde testeábamos entidades, controladores y servicios previamente creados. En ellas, probábamos todos los métodos y funciones, donde en cada prueba dábamos valores ciertos o falsos para probar el buen funcionamiento de ese método en concreto. Por ejemplo, si una función debía devolver un valor erróneo con una entrada específica, en la prueba pásabamos como parámetro dicho valor para comprobar que la restricción del método funcionase correctamente.

Para las **pruebas de vista**, se ha utilizado en DP1 MockMVC para comprobar tanto vistas, como envíos de formulario en POST. En EGC se han implementado pruebas haciendo uso de scripts en Python con Selenium. Fuera de la asignatura se han realizado pruebas de vistas con Electron y NodeJs por parte de algún integrante. Para las **pruebas de carga**, se ha utilizado locust en EGC, y alguna otra herramienta online para medir métricas de calidad en PGPI.

También, conocemos a menor escala la **cobertura de pruebas.** Esto es el porcentaje de pruebas realizadas en un proyecto concreto. Es decir, a mayor cobertura de pruebas, mayor código probado y por tanto, menor probabilidad de fallo en el proyecto.

Bibliografía

Intencionadamente en blanco.