C1.04.06

Conocimiento sobre testing

Repositorio: https://github.com/C1-04-06/Acme-L3-D01.git

Fecha: 14 de febrero de 2023

Integrantes

Diego García Linares diegarlin@alum.us.es
María Márquez Soldán marmarsol4@alum.us.es
Juan Carlos Morato Navarro juamornav3@alum.us.es
Olegario Morato Navarro olemornav@alum.us.es
Úrsula Garrucho Sánchez ursgarsan@alum.us.es

ÍNDICE

Tabla de revisiones

Nº	Fecha	Descripción
1	14/02/2023	Elaboración del report
2	16/02/2023	Corrección tras la sesión de seguimiento

Introducción

Este reporte contiene los conocimientos previos que el grupo posee en cuanto a testing del WIS (*Web Information System*) en base a asignaturas similares cursadas con anterioridad, principalmente DP1.

Contenido

Conocimientos previos sobre testing

Un test en nuestro contexto, son una serie de pruebas que verifican que las funcionalidades del sistema funcionan correctamente. Gracias a ellos, se pueden encontrar fallos o bugs en el sistema. Un **fallo** es un problema generado por un error en la elaboración del código de una funcionalidad del sistema, el cuál provoca un resultado no deseado. Un **bug** es un fallo no esperado en el código que provoca resultados no deseados para el sistema.

Respecto a testing, se tiene conocimiento del concepto de SUT gracias a DP1, representando aquella parte de código sobre la cual se realizan pruebas. Una prueba consta en general de 3 partes:

- **Arrange/Fixture:** aquí se inicializan los datos de prueba, sobre los cuales se conoce el resultado de la ejecución.
- **Act:** se corresponde con la ejecución del método o métodos a probar
- **Assert:** son comprobaciones de que los resultados de la ejecución y los esperados son iguales.

También se conoce que las pruebas se localizan en un rango según su granularidad. Estos son clasificados de menor a mayor granularidad como:

Unit tests: Se caracterizan por ser test que se enfocan en partes pequeñas del sistema para detectar posibles errores en funcionalidades de servicios que se ofrecen. El número de test que se realizan depende de lo exhaustivo que se quiera llegar a ser, aunque normalmente se recomienda intentar cubrir los casos más importantes o más comunes. Para ello se crea un camino **positivo,** el cual busca que el servicio del sistema obtenga resultados esperables y que actúe de forma correcta, y por otro lado se tiene el camino **negativo,** cuyo objetivo es comprobar que el sistema responde bien en caso de errores dependiendo del contexto del problema.

Integration tests: el objetivo en este tipo de tests es probar la comunicación entre los componentes que forman el sistema, ya sea tanto en ámbito de software como hardware.

End-to-end tests: se trata de una metodología para garantizar que las aplicaciones se comportan como se espera en un escenario real, simulando una situación de uso dependiendo del contexto de la aplicación.

Acceptance tests: son las pruebas que determinan si se han cumplido las necesidades o requisitos del cliente.

Exploratory tests: principalmente es usado para el aprendizaje del tester manejando el sistema mientras crea las pruebas, y de esta manera se generan nuevas pruebas conforme se aprende del sistema.

A la hora de crear tests, se puede hacer uso de dobles, estos son datos que simulan a los reales cuya finalidad es ser usados en los test para no trabajar con los datos reales de la base de datos o incluso con funcionalidades de repositorios y servicios. Algunos tipos de dobles son los Stubs, Mocks y Fakes.

Conclusiones

El equipo ya poseía conocimientos sobre testing, principalmente sobre pruebas unitarias y un poco de pruebas de integración. Se espera que durante el desarrollo de la asignatura podamos ampliar estos conocimientos mediante las herramientas que nos propone la asignatura.

Bibliografía

Intencionadamente en blanco.