INFORME DE LAS PRUEBAS DE UN WIS

GRUPO: C2.02.11

Francisco Javier de la Prada Prados (fraprapra1@alum.us.es)
Pablo Quindós de la Riva (pabquide@alum.us.es)
María José Ruiz Vázquez (marruivaz1@alum.us.es)
Juan Luis Ruano Muriedas (juaruamur@alum.us.es)
Santiago Zuleta de Reales Toro (santizuleta11@gmail.com)

Índice

Resumen ejecutivo

En este reporte vamos a detallar cuales son los conocimientos de los integrantes del grupo acerca de cómo realizar las pruebas a un WIS (Web Information System) que hemos adquirido y poseemos a la hora de cursar esta asignatura.

Historial de versiones

Fecha	Versión	Descripción de los cambios	Sprint
17/02/2023	1.0	Creación de los documentos para el entregable 1	1

1. Introducción

Nuestro conocimiento sobre las pruebas de un WIS provienen principalmente de las asignaturas de Ingeniería del Software donde algunas de las asignaturas nos enseñan los principios básicos, reglas y hasta los tipos de pruebas que existen hasta entrar un poco más en detalle sobre ellas a modo práctico.

Para ellos vamos a enumerar dichas asignaturas, así como el curso donde las cursamos, para posteriormente describir qué aprendimos sobre cómo testear un WIS.

2. Conocimientos previos sobre las pruebas de un WIS

2.1 Arquitectura e Integración de Sistemas Software

Las pruebas no permiten garantizar que un programa no contiene errores. No se puede probar todo el sistema ya que hay muchas combinaciones posibles, pero se deben probar los casos más comunes.

Sirven para detectar el mayor número posible de errores con poco tiempo y esfuerzo.

Hay dos tipos de pruebas: funcionales (buscar errores relacionados con la funcionalidad del sistema) y no funcionales (rendimiento, seguridad, usabilidad, fiabilidad).

Tenemos las pruebas unitarias (prueban una parte pequeña y específica de un programa), pruebas de integración (probar la interacción entre los distintos componentes de un sistema), pruebas de sistema (probar el sistema en su totalidad) y pruebas de aceptación (verifican que el sistema cumple con los requisitos de usuario).

2.2 Diseño y Pruebas I

Las pruebas que se pueden hacer a un WIS son validación y verificación, es decir, comprueban que las unidades de código se comporten como se espera según los requisitos.

Cada prueba comprueba una parte del sistema (System Under Test). Un conjunto de pruebas es un test suite. Hay que testear todo lo que podamos, probar distintas situaciones, tanto las positivas como las negativas, para probar que el sistema responde como se desea y automatizar las pruebas de manera que sean rápidas de aplicar. Estas propiedades son las llamadas “First”, proveniente del inglés “Fast”, “Independent”, “Repeatable”, “Self-checking”, “Timely”.

Existe un framework de pruebas llamado JUnit 5, el cual nos permite configurar los datos de prueba (fixture), ejecutarlas sobre el sujeto bajo prueba (Act) y comprobar que el resultado obtenido por esa prueba es el que se ha devuelto (Assert).

3. Tipos de pruebas

A continuación vamos a nombrar algunos tipos de pruebas que conocemos para complementar la información.

- **Pruebas unitarias**

Sujeto bajo prueba es una unidad. La mayoría de las pruebas en una suite de pruebas deben ser unitarias. Se configura el dato a probar, se corre la prueba y se afirma que el resultado es el esperado. Este fue el tipo de prueba más trabajado en la asignatura.

- **Pruebas dobles**

Objeto que sustituye a la producción de objetos con una versión más rápida, simple y ficticia. Permiten probar el SUT aisladamente.

- **Pruebas sociables**

Permiten la comunicación real con los colaboradores. Asume que todas las demás pruebas funcionan perfectamente. Usan pruebas dobles solo si los colaboradores son lentos.

- **Pruebas solitarias**

Los colaboradores del SUT se sustituyen por dobles. Tienen un aislamiento perfecto.

- **SEAMS**

Lugar donde puedes alterar el comportamiento en tu programa en tu programa sin editarlo. Los SEAMS son pruebas dobles que aíslan el comportamiento del SUT de los comportamientos de otros colaboradores.

- **STUBS**

Se usa cuando se necesita un controlador para devolver un valor específico para llevar el SUT a un estado concreto. Comprobamos dicho estado para decidir si pasa la prueba.

- **MOCKS-SIMULACROS**

Se usa para probar interacciones entre objetos. Es útil cuando no hay cambios de estado en el SUT. Verificamos el comportamiento en el SUT para decidir si pasa la prueba.

- **FAKES-FALSIFICACIONES**

Implementaciones ligeras de API que se comportan como las reales, pero no estan listas para producción. Se usa cuando la implementación real no se puede usar en las pruebas. Normalmente los implementa el mismo equipo.

- **DUMMY-FICTICIAS**

Objetos que pasan las pruebas pero nunca se utilizan. Se suelen usar para completar listas de parámetros.

Además, para complementar los test, tenemos las siguientes herramientas:

- **Mockito**

Framework simulador para pruebas unitarias.

- **@WebMvcTest**

Anotaciones que sirven para probar controladores, ya que las pruebas unitarias de los controladores no son 100% solitarias.