TESTING WIS

Grupo E7.02

Álvaro Úbeda Ruiz (alvuberui@alum.us.es)

Mario Pérez Coronel (marpercor8@alum.us.es)

Carlos Garrido Rodríguez (cargarrod12@alum.us.es)

Ramón Rodríguez Bejarano (ramrodbej@alum.us.es)

Mario Rodríguez García (marrodgar62@alum.us.es)

Juan Carlos Gómez Rodríguez (juagomram4@alum.us.es)

Repositorio: https://github.com/alvuberui/Acme-Toolkits

Tablero: https://github.com/users/alvuberui/projects/5

31 DE MAYO DE 2022

1. Resumen ejecutivo

El presente documento explicaremos los conocimientos que hemos adquirido tras la realización de la asignatura Diseño y Pruebas II complementando así los conocimientos ya obtenidos en asignaturas como Diseño y Pruebas I, Arquitectura e Integración de Sistemas Software o Introducción a la Ingeniería del Software y los Sistemas de Información.

2. Tabla de revisión

Num. Revisión	Fecha	Descripción
1	31/05/2022	Creado el documento y apartados 1 y 3.
2	31/05/2022	Realización del apartado 4.
3	31/05/2022	Realizada la conclusión 2
4	1/06/2022	Revisión final antes de la entrega.

3. Introducción

Como bien ya se ha explicado en el apartado uno “Resumen ejecutivo” este documento tiene como objetivo explicar qué conocimientos hemos adquirido en la asignatura Diseño y Pruebas II una vez hemos terminado, en cuanto a *testing* se refiere.

4. Framework

A lo largo de la asignatura hemos implementado test con la ayuda del *Framework* de la asignatura. Este *Framework* tiene implementado a través de diferentes métodos una manera fácil de testear nuestra aplicación que haya sido implementada con las mismas herramientas que nos proporciona para desarrollar las diferentes funcionalidades.

A parte de poder hacer distintos test parametrizados, también nos permite crear diferentes métodos para poder implementarlo en nuestros test individuales.

Todo esto no sería posible si el Framework no tuviera integrado el modo marionete, un modo de Firefox que junto con geckodriver, un driver que nos proporciona el control total de una pestaña de Firefox, ejecuta los test de forma automatizada con los datos de ejemplo que son utilizados y proporcionados a la aplicación para crear los diferentes escenarios para los distintos test.

5. Implementación:

Como ya se ha visto también en el documento de “Arquitecture WIS”, ahora vamos a describir como hemos realizado los diferentes test para las funcionalidades implementadas en “Acme-Toolkits”.

```java
public class InventorArtefactCreateTest extends TestHarness {

    @ParameterizedTest
    @CsvFileSource(resources = "/inventor/artefact/create-positive.csv", encoding = "utf-8", numLinesT
    @Order(10)
    public void positiveTest(final int recordIndex, final String type, final String name, final String
        super.signIn("inventor1", "inventor1");

        super.clickOnMenu("Inventor", "Own artefacts");
        super.checkListingExists();

        super.clickOnButton("Create");
        super.fillInputBoxIn("type", type);
        super.fillInputBoxIn("name", name);
        super.fillInputBoxIn("code", code);
        super.fillInputBoxIn("technology", technology);
        super.fillInputBoxIn("description", description);
        super.fillInputBoxIn("retailPrice", retailPrice);
        super.fillInputBoxIn("moreInfo", moreInfo);
        super.clickOnSubmit("Create");

        super.checkListingExists();
        super.sortListing(2, "desc");

        //Revisar que la cabecera esta bien
        super.checkColumnHasValue(recordIndex, 0, type);
        super.checkColumnHasValue(recordIndex, 1, name);
        super.checkColumnHasValue(recordIndex, 2, code);
```

Aquí podemos ver una captura de uno de nuestros tests. Primero, comentar que dentro de estas clases podemos hacer tests positivos donde comprobamos que la ejecución del mismo se realiza correctamente (como la captura adjuntada) y también hacemos tests negativos donde comprobamos que algo ha ido mal ya sea porque no tengamos autorización, o para comprobar una validación entre otras cosas. Dentro de estos tests hemos aprendido a usar *TestHarness* , que es el paquete importado que hemos utilizado, y sus funciones con super para rellenar input-box, comprobar que existen listados, *logearnos* con un Rol que queramos, comprobar que el propio test ha fallado si es negativo, y un sinfín de posibilidades.

Como podemos ver en el ejemplo el test tiene distintas partes.

Por un lado, tenemos *@ParametirzedTest* junto a *@CsvFileSource* que son dos atributos de que nos permite proporcionar al test distintos datos parametrizados desde un archivo "*.csv*", de esta forma tenemos por un lado el test y por otra los datos que vamos a comprobar.

Por otro lado, el test extiende a la clase *TestHarness* que es una clase de ejemplo creada por los profesores para introducirnos una forma de heredar distintos métodos que se van a repetir durante la ejecución de los test. En este caso el método *signIn(String, String)* no está implementado en el *Framework* pero si esta clase.

Y ultimo tenemos los métodos implementados por el framework que para nosotros podrían ser los métodos más primitivos ya la mayoría de las veces con estos métodos podría crear nuestros test sin ningún problema. Por ejemplo: "*clickOnButton*" o "*fillInputBoxIn*".

6. Conclusión

A lo largo de esta asignatura hemos afianzado más los conocimientos, algo escasos, que teníamos sobre el testing ya que, como comentamos en un documento con el mismo nombre que entregamos al principio del cuatrimestre, en asignaturas como AIS, IISI o DP1 habiamos aprendido sobre testear algunos de los proyectos que implementamos pero de una manera superficial.

En este caso por primera vez hemos intentado testear las funcionalidades básicas o esenciales de la aplicación. Utilizando las herramientas que nos proporcionan hemos implementado test automatizados que nos permiten saber si las funcionalidades de la aplicación están teniendo resultado.

Como conclusión final, tras acabar el proyecto y el cuatrimestre hemos sido capaz de implementar los métodos necesarios para comprobar que nuestra aplicación funcione con normalidad tras un cambio en el modelo o un cambio en las funcionalidades.

7. Bibliografía

Intencionadamente en blanco.

.