WIS Testing Report

Group: E7.05

Repository: https://github.com/PabloAlvarezCaroUS/Acme-Toolkits

Student #1

ID Number: 29513104X

Name: Álvarez Caro, Pablo

Roles: Manager, Developer

E-mail: pabalvcar@alum.us.es

Student #2

ID Number: 77863065S

Name: García Bohórquez, José Ignacio

Roles: Developer, Tester

E-Mail: josgarboh@alum.us.es

Student #3

ID Number: 53963760X

Name: Marín Gómez, Pablo

Roles: Developer, Tester

E-Mail: pabmargom3@alum.us.es

Student #4

ID Number: 29541858Z

Name: Migueles Dominguez, Francisco Javier

Roles: Analyst, Developer

E-Mail: framigdom@alum.us.es

Student #5

ID Number: 49128512Y

Name: Ruiz Gil, Diego

Roles: Developer, Tester

E-mail: dieruigil@alum.us.es

Student #6

ID Number: 29544227Z

Name: Zamora Fernández, David

Roles: Developer, Tester

E-Mail: davzamfer@alum.us.es

Date: 28/05/2022

Índice

Historial de versiones

Versión	Fecha	Registro de cambios
1.0	23/05/2022	● Versión inicial
2.0	28/05/2022	● Completado

Resumen Ejecutivo

En este documento vamos a exponer los conocimientos adquiridos respecto a la realización de pruebas de una WIS a lo largo de la asignatura.

Introducción

Comentaremos los distintos aprendizajes adquiridos a lo largo del proyecto, hablando, además, sobre la toma de contacto por parte de los integrantes del “Modo Marioneta” y el método de trabajo con archivos csv.

Contenido

En esta asignatura, hemos aprendido a hacer testing E2E, haciendo uso de diversas herramientas, entre ellas, Firefox como navegador para realizar las pruebas y geckodriver para automatizar estas.

Modo Marioneta

Todos los integrantes del grupo desconocíamos la existencia de la funcionalidad “modo marioneta” de Firefox, la cual resulta muy útil para poder visualizar cómo se realizan los tests, y en caso de fallo tener más claro el porqué y el dónde, agilizando el proceso de arreglar dichos tests.

Este modo marioneta también nos permite probar de forma más precisa la aplicación final que va a recibir el usuario, a coste de un mayor tiempo de testeo.

Hemos trabajado con el modo marioneta realizando test que extienden de una clase del framework llamada “TestHarness”, que a su vez, extiende de un “AbstractController” que interactúa con GeckoDriver para manipular el comportamiento de Firefox durante los tests.

Datos y métodos de prueba

También hemos aprendido a hacer testing con diferentes archivos “.csv” en función de qué tipo de método de prueba se trate (métodos positivos o negativos, métodos de crear, mostrar, o borrar…).

Estos datos los recibirá el test como cadenas de texto que usará para interactuar con la página, ya sea creando un elemento nuevo, editándolo… Finalmente, realiza una comprobación/comparación de la información del csv con la información introducida en la página.

En el caso de testeo de show y list, el primer paso antes citado no se realiza, la clase de test solamente se encargaría de comprobar que se ha mostrado o listado correctamente, según corresponda

Además desconocíamos algunos métodos necesarios en la asignatura, como es el caso de algunos métodos de “hacking”, que refuerzan la aplicación probando el acceso de distintas autoridades en las páginas existentes. Esto normalmente se realiza navegando hacia urls a las cuales no debería de tener permisos o bien por su autoridad, lo cual, suele encargarse el framework, o bien por otros casos de los cuales nos hemos encargado nosotros.

Conclusión

En conclusión, hemos aprendido nuevas formas de probar nuestro código, como es el caso de los tests automáticos que usan “marionette” para probar la interfaz de nuestra WIS. Además, hemos adquirido una cultura de realizar tests para las funcionalidades que realizamos, para asegurarnos completamente de su correcto funcionamiento.