Grupo 01 - Pruebas de Software

Link Video: https://youtu.be/YNRki0h2waY

Descripción de la entrega 3:

Esta entrega se centra en la integración de Selenium como herramienta para realizar pruebas automatizadas en la interfaz de usuario. El objetivo principal es mejorar la calidad del proyecto mediante la detección temprana de errores cada vez que se realicen cambios en el repositorio.

Al automatizar las pruebas, se busca minimizar los errores, garantizando que el software cumpla con los requisitos establecidos y que esté alineado con las expectativas de los usuarios y las necesidades del negocio. Esta integración también contribuye a mantener un desarrollo más ágil y confiable, reduciendo el tiempo y esfuerzo necesario para validar cambios.

Uso de Selenium:

Selenium es un entorno de pruebas de software diseñado para aplicaciones web en múltiples navegadores. Su principal ventaja es la capacidad de detectar problemas en la interfaz del software, permitiendo verificar cómo interactúan los usuarios con el sistema y asegurándose de que la interfaz pueda manejar errores y cumplir con las funcionalidades esperadas.

En este proyecto, se realizaron cinco pruebas automatizadas con Selenium para validar los principales requerimientos de la aplicación. Estas pruebas son:

- Inicio de sesión usuario o Administrador (IngresarUsuario.js y IngresarAdmin.js): Verifica que la interfaz permite iniciar sesión correctamente tanto para usuarios como para administradores.
- Agregar producto como Administrador (AgregarProducto.js): Comprueba la funcionalidad de agregar un producto llamado "Producto de Prueba" como administrador, asegurándose de que la acción se realice correctamente.
- Editar Producto como Administrador (EditarProducto.js): Edita el producto "Producto de Prueba", modificando todos sus campos como administrador, para verificar que la funcionalidad de edición opera correctamente.
- Eliminar Producto como Administrador (EliminarProducto.js): Elimina el producto previamente editado, "Producto de Prueba Editado", y confirma que la funcionalidad de eliminación se ejecuta sin errores, asegurándose de que no queden rastros del producto.

Integración con Jenkins: En esta entrega integramos las pruebas hechas en Selenium como una etapa dentro del Pipeline, con el fin de garantizar el funcionamiento de las operaciones descritas previamente. Estas pruebas, a la par de los test realizados al Backend, es necesario que sean exitosas para lograr el despliegue de la aplicación, ya que si no es así, significa que estaríamos desplegando un producto defectuoso al incluir nuevos cambios a la rama main.

Notificaciones Slack:

En el pipeline se implementa el envío de notificaciones a Slack en el canal del repositorio.

- En el caso de que la ejecución sea fallida, se envía el mensaje “El pipeline falló, se detuvieron todos los procesos, se sugiere revisar los registros generados.”
- En el caso de que la ejecución sea exitosa, se envía el mensaje “El pipeline ha sido exitoso, se han desplegado los nuevos cambios.”

Problemas y Soluciones:

En la implementación de Selenium en el pipeline, encontramos dos problemas principales:

- Al implementar Selenium de forma local y ejecutar las pruebas no tuvimos ningún inconveniente, pero al ejecutar las pruebas en la máquina virtual de AWS, las pruebas tardaban en ejecutarse y no se ejecutaban correctamente, arrojando un error de ejecución. La solución a este problema fue agregando tiempos de Sleep, para que la ejecución de Selenium se realice de mejor forma y más controlada.
- Al ejecutar las pruebas de Selenium en la máquina virtual de AWS, al parecer no se ejecutaba correctamente el navegador Chrome de las pruebas, lo cual arrojaba un error de ejecución. La solución a este problema fue encontrada en el foro de “stackoverflow”, en donde se agrega el flag de ejecución “--no-sandbox” en cada prueba integrada de Selenium. Luego el pipeline se ejecuta sin problemas.