**Taller Selenium Web Driver**

**Testing y Aseguramiento de la Calidad**

**Integrantes**

* Jeisson G. Sánchez
* Daniel Walteros
* Orlando A. Gelves Kerguelen
* Brayan Alexis Jiménez Moya
* Juan Sebastián Frásica Galeano

**Descripción de la plataforma a probar:**

Con motivos del presente trabajo se determinó utilizar una aplicación que fue implementada como proyecto de otra asignatura, para la gestión y administración de libros y prestamos dentro de la biblioteca de la escuela, esta aplicación fue escrita en Java y desplegada sobre Heroku.

URL de la aplicación: <http://ecibrary.herokuapp.com/>

**Consideraciones al momento de construir las pruebas:**

Aunque la aplicación fue escrita completamente en java, tuvimos que considerar el hecho de que las librerías de JSF (Java Servlet Faces), y Primefaces para algunas características graficas de la aplicación implementan por debajo contenido JavaScript que no se encuentra disponible recién se abre la aplicación, ya que este requiere un procesado para posteriormente ser renderizado dentro del sitio web.

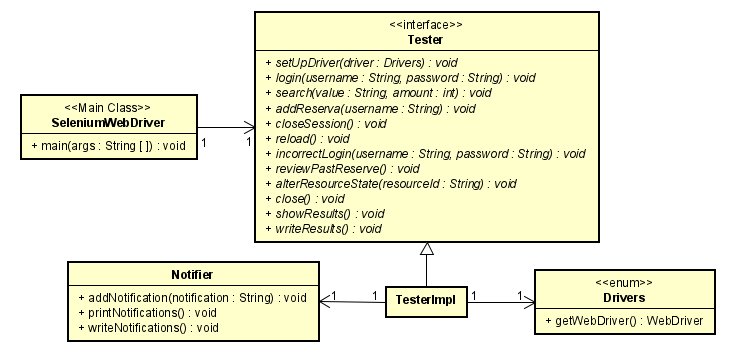
Para ello tuvimos que plantear un sistema de delay para acceder a los ítems del menú de navegación y eventos sobre botones.

**Estructura del proyecto Java**

Dentro del proyecto se decidió por fines prácticos modularizar la lógica para gestionar los drivers dentro de la aplicación, y definir una interfaz que nos definiera las características necesarias para las pruebas llamada Tester y la respectiva clase TesterImpl que se encarga de implementar la lógica requerida dentro del proyecto para que efectivamente las pruebas funcionen.

Se definió también un enum de driver el cual nos sirve con el fin de gestionar los diferentes drivers de los diferentes browsers sobre los cuales podemos correr respectivamente cada una de las pruebas automatizadas, esto fue realizado con el fin de poder probar la interoperabilidad entre diferentes browsers de forma más dinámica y simple.

Para mostrar los resultados de la ejecución de las distintas pruebas se desarrolló una clase Notifier la cual nos permite al finalizar las pruebas, mostrar mediante consola o en un archivo de texto los resultados de la ejecución de las distintas pruebas planteadas en el aplicativo.

[](https://github.com/Silenrate/TACS-SeleniumTest/blob/master/diagram/class.PNG)

**Casos de Prueba:**

* **Login**: Validamos que un usuario previamente registrado este en capacidad de iniciar sesión satisfactoriamente.
* **Login incorrecto**: Validamos que se mostrara un mensaje de error en caso de que se llevara a cabo un login con credenciales incorrectas (inexistentes).
* **Búsqueda de recurso por nombre**: Con un usuario autenticado, filtramos la búsqueda de recursos por nombre y validamos que los cinco primeros resultados tuvieran en su nombre el valor por el cual filtramos.
* **Solicitud de préstamos**: Validamos que, para el usuario previamente autenticado, se pudiera registrar satisfactoriamente el prestamos de materia editorial de la escuela, y efectivamente dicho préstamo pueda ser evidenciado dentro del calendario de la aplicación.
* **Reserva pasada**: Se revisa que se quedó guardada una reserva pasada, se verifica la fecha así se guarda un registro de log para su posterior verificación, la hora esperada debe ser igual a la que está en el sistema
* **Cambiar estado de un recurso**: Se busca cierto recurso, y se le cambia su estado de “Disponible” a “Mantenimiento”. Luego, se valida que el cambio fue hecho y se revierte el cambio para mantener la integridad de los datos.

**Información necesaria:**

* Para poder ejecutar las pruebas se requieren credenciales de acceso de un usuario normal y de administrador a la plataforma, ya que esta aplicación no posee sistema de registro, estas credenciales nos fueron suministradas por uno de los desarrolladores de la aplicación.

**URL del proyecto:**

<https://github.com/Silenrate/TACS-SeleniumTest>