TESTING PARA LA FORMACIÓN DE QA:

Módulo: Proyecto integrado

QA TRAINING

TFG del ciclo de Desarrollo de Aplicaciones Web (DAW)

Curso 2022-2023

Joan Facundo Tamborini Cabral

Jueves, 27 de Mayo de 2023

Abstract

This project has a very clear orientation: its main issue is to reach students who are starting to learn about automation testing in order to become quality assurance (QA) inside a software company, as well as those workers, either on school or inside a company itself, who are willing to, or just get pushed to, teach those skills to others.

Following this way of thought, this project is lead to give a complete, easy to handle and compeling tool to use technologies such as Postman, Selenium or Jmeter, launching on it every kind of tests, from performance to functional.

Resumen

Este proyecto tiene un objetivo claro: su tarea principal es llegar a aquellos estudiantes que están empezando a aprender sobre la automatización de tests buscando convertirse en quality assurance (QA, gestión de calidad) dentro de una compañía de software, así como a aquellos trabajadores, tanto de centros formativos como de empresas propiamente dichas, que desean, o se ven obligadas, a enseñar estas habilidades a otras personas.

Siguiendo esta línea de pensamiento, este proyecto lleva a una herramienta sencilla, completa y competitiva para utilizar tecnologías como Postman, Selenium o Jmeter, para realizar tests de todo tipo, desde funcionales hasta de rendimiento.

Tabla de contenido

1. Justificación del proyecto.

La idea de este proyecto surge de las necesidades encontradas en el sector del testing de software al comenzar mi Formación en Centro de Trabajo (FCT).

Mientras que para el aprendizaje de personal de desarrollo es sencillo encontrar una gran cantidad de información y medios, en el ámbito del testing, a la hora de realizar pruebas para el aprendizaje de herramientas necesarias para un Quality Assurance (en adelante, QA), tales como Jmeter, Postman, Cypress o Selenium, resulta difícil encontrar páginas que permitan realizar las interacciones que se buscan, como un log in en un sitio web que redirija a distintas páginas del mismo o la localización de elementos dinámicos que permitan realizar pruebas de mayor complejidad.

Para solventar esto, mi intención en este proyecto es la realización de un sitio web que cubra dichas necesidades, con unos objetivos y un target de usuarios muy claro que se ha mencionado ya, y que se detallará a lo largo de este documento.

El enfoque un tanto diferente de este proyecto hace que algunas partes del mismo difieran de lo que suele ser habitual, como la planificación, que no constará de diagramas de caso de uso ni de entidad relación, ya que los primeros carecen de sentido al no existir una manera concreta en la que el usuario deba interactuar con la aplicación, si no que cada usuario decidirá, mediante sus pruebas automatizadas, los pasos que seguirá y cómo navegará por los distintos apartados.

En cuanto a los diagramas de entidad relación, y de nuevo por las particularidades del proyecto, la base de datos es tan simple como una tabla que contiene datos de tres tipos, como se mostrará en próximos apartados.

Por todo esto, y como se verá conforme se avance en este documento, algunos puntos importantes en otros proyectos se han omitido en este, al igual que algunos que no suelen acompañar a los proyectos se incluirán en él.

2. Introducción.

El proyecto Testing para la formación de QA se desarrollará teniendo en mente un objetivo muy claro ya mencionado, y que indica claramente el nombre del mismo: facilitar la formación de nuevo personal de QA.

Este es un ámbito del desarrollo de software que en numerosas ocasiones es ignorado, o pasado por alto, hasta el punto de que no existían, hasta hace relativamente poco, teniendo en cuenta la historia de las tecnologías de la información, herramientas que facilitasen este proceso. Además, durante mucho tiempo, es un trabajo que se ha cargado a las espaldas del propio desarrollador, personal que realmente no está cualificado para llevarlo a cabo y que, además, limita aún más el ya escaso tiempo para cumplir los plazos de entrega del desarrollo de una determinada aplicación.

a. Entornos en desarrollo de software.

Sin embargo, a día de hoy, y principalmente en medianas y grandes empresas, esto está cambiando, y cada vez más se busca tener un equipo especializado en probar el software a lo largo de los cinco entornos por los que este pasa durante su desarrollo:

- Entorno de desarrollo. Esta es la etapa en la que cada programador trabaja de manera local con una copia del proyecto. Aquí pueden surgir errores de código, muy comunes ya que se estima que cada programador comete un error por cada 30-50 líneas escritas. Estos errores, por lo general, los detectará el propio programador, ya que, normalmente, será la única persona en contacto con esa parte del código.

- Entorno de integración. Aquí, las distintas ramas que se han generado para que cada programador trabaje por su cuenta se juntan, generando, por lo general, un gran número de errores de integración. Estos se detectan al intentar ejecutar las funcionalidades que se han desarrollado por separado y que, por este mismo motivo, suelen contener irregularidades en su contenido, tales como variables con nombres repetidos o llamadas a funciones con la nomenclatura equivocada. Si la ejecución tras la integración está a cargo de personal de QA, este podrá realizar un informe que entregar al encargado del proyecto para que revisen todos los fallos localizados.

- Entornos de pruebas. Es el principal entorno del QA, en el que debe utilizar la aplicación móvil o sitio web como si de un usuario se tratase, localizando en el proceso tantos errores como le sean posible. Aquí es también donde se pueden empezar a emplear técnicas de automatización que faciliten esta tarea y el futuro mantenimiento.

- Entorno de pre-producción. Se trata de un entorno en el que se trabaja con las condiciones más similares, a poder ser idénticas, a las que tendrá el software cuando se ponga a disposición del cliente. Aquí, la automatización de pruebas realizada por el QA en el anterior entorno se utiliza para comprobar que, bajo las condiciones previstas, el software funciona como se esperaba al comenzar su desarrollo.

- Entorno de producción. Es el entorno real en el que se ejecuta la aplicación desarrollada. Como labores de mantenimiento, es probable que el cliente solicite que las pruebas sigan realizándose para poder localizar errores tan pronto como surjan, y esto es algo que permite y facilita en gran medida la automatización de pruebas.

b. Habilidades de un QA.

Como se puede apreciar, la detección de errores es una parte fundamental del desarrollo software, y está ligado a cada una de las partes del proceso. Es por esto que tener personal especializado en este ámbito, con las habilidades adecuadas y las herramientas necesarias a su disposición, es vital para llevar el proceso a buen puerto.

Entre estas habilidades mencionadas, destacan las siguientes:

- Comprender el ciclo de desarrollo de un software. Al estar en estrecha relación con las personas encargadas del desarrollo de la aplicación web, es fundamental que conozca los diferentes pasos que se darán para poder adaptar su punto de vista al tipo de errores que se buscan en cada momento, ya que varían conforme el proyecto avanza.

- Comunicación interpersonal. El personal de QA debe comunicarse con personas entendidas en la materia de la programación, pero también con otras totalmente ajenas a este, por lo que su habilidad para comunicar la información tanto de manera técnica como con un lenguaje más cotidiano es de vital importancia.

- Resolución de conflictos. Esto es algo que tienen en común todas las personas dedicadas a un proyecto de desarrollo de software, puesto que el trabajo en equipo es parte indispensable. Generar un ambiente de trabajo agradable sin por ello dejar de ofrecer siempre la opinión personal aunque choque con la de otro miembro del grupo marcará la diferencia en un buen QA.

- Planificación y organización (gestión del tiempo). Debido a que los desarrollos tienen una fecha de entrega (deadline) definida por el cliente, que no siempre responde a las necesidades reales de tiempo para su finalización, gestionar el tiempo planificando las pruebas y organizándolas en los distintos momentos del desarrollo es un punto clave.

- Atención al detalle. Del mismo modo que un programador debe acostumbrar su ojo a ver ese paréntesis que no se cerró donde debía o esa coma que falta, el QA debe aprender a evaluar rápidamente cada interacción posible en una aplicación web o móvil, pudiendo así idear pruebas que pongan a prueba los límites de la misma para asegurar el buen funcionamiento del producto.

c. Herramientas del personal QA.

También es importante mencionar las herramientas que se han tenido en mente a la hora de desarrollar este proyecto. Como ya se ha mencionado, la automatización de pruebas está en pleno auge, por lo que pueden encontrarse una gran cantidad y variedad de herramientas, cada una con sus características propias y destinadas a un uso en concreto. Sin embargo, y como suele ser habitual, algunas han ido destacando del resto y formando las más socorridas por el personal de QA:

- Jmeter. Es la herramienta de pruebas de rendimiento por excelencia. A través de una interfaz gráfica permite, de una manera muy sencilla, realizar, principalemte, tres tipos de pruebas:
    - Pruebas de carga (load testing). Estas pruebas consisten en aumentar la carga de trabajo de la aplicación mediante la adición de un número de usuarios concurrentes cada vez mayor.
    - Pruebas de estrés (stress testing). En estas, se pone a prueba el límite superior admitido por la aplicación en cuanto a carga de trabajo: se sobrepasa el límite en un momento puntual para comprobar cómo responde a esto.

    - Pruebas de resistencia (endurance testing). Consisten en generar unas condiciones de carga de trabajo constantes pero prolongadas en el tiempo. Son semejantes a las pruebas de carga, solo que aplicadas durante un lapso de tiempo mayor.

  Cabe destacar que todos los datos obtenidos por los distintos tipos de pruebas que permite Jmeter se incluyen en informes que el usuario puede personalizar para mostrar aquella información que consideren más relevante en función del proyecto.

- Postman. Esta herramienta, mediante métodos como GET, POST, PUT o DELETE, realiza interacciones con una API, pudiendo probar si la respuesta que esta ofrece a los métodos citados es la correcta. El método GET, por ejemplo, permite recibir, en un archivo JSON, la información contenida en la API que coincida con las condiciones dadas, comprobando así si la respuesta HTTP a las peticiones realizadas es la correcta.

- Selenium. Está compuesto por un conjunto de utilidades que permiten crear pruebas de aplicaciones web. Además, resulta muy versátil al poder ser utilizado en diversos lenguajes de programación (Java, JavaScript, python, C#, etcétera). También permita realizar capturas de pantalla de los tests, para ser visualizadas posteriormente y comprender mejor los fallos que se produzcan al realizar las pruebas.

- Cypress. Surge como evolución de Selenium, ofreciendo, a diferencia de la anterior, una interfaz gráfica amigable que facilita la labor del tester, así como la comprensión de los resultados, al estar toda la información recibida de las pruebas en un formato mucho más sencillo de entender. Es una herramienta bastante nueva pero que ha cogido fuerza intentando suplir las carencias de Selenium.

- Appium. Aunque no tiene relación con el proyecto, al tratarse de un software destinado a pruebas en aplicaciones móviles, merece la pena mencionarlo, ya que es la herramienta más utilizada para este propósito. Trabajando con un dispositivo real o con una virtualización creada, por ejemplo, en Android Studio, permite realizar todo tipo de pruebas automáticas sobre cualquier aplicación web.

3. Objetivos.

Los objetivos de este proyecto, ya mencionados anteriormente pero que se detallan a continuación, se desglosan en dos tipos:

a. Objetivos generales.

El principal objetivo es que el usuario final del software a desarrollar pueda realizar pruebas de todo tipo con sus herramientas de testing de software. Esto incluye tanto las pruebas de rendimiento como las de funcionalidad. De este modo, en un único portal podrá poner en práctica todo lo aprendido en su formación como QA, ahorrando tiempo en buscar diferentes sitios web que le permitan realizar una u otra prueba.

b. Objetivos específicos.

Por otra parte, el proyecto contará con varios objetivos específicos, los cuales se irán superando a lo largo de todo el desarrollo del proyecto:

1. Investigar las capacidades de Selenium Webdriver y familiarizarse con sus comandos y nomenclatura:

    - Realizar una revisión exhaustiva de la documentación oficial de Selenium.

    - Explorar los conceptos fundamentales de Selenium, como los localizadores, las acciones del navegador y las aserciones.

    - Aprovechar la experiencia adquirida en la Formación en Centros de Trabajo de esta herramienta de testing.

2. Implementar casos de prueba automatizados para un conjunto de funcionalidades clave de la aplicación web:

    - Identificar las distintas funcionalidades y utilizar Selenium para automatizar su interactividad.

    - Diseñar casos de prueba efectivos y completos que cubran las diferentes rutas de la aplicación web.

    - Utilizar las capacidades de Selenium para escribir pruebas que reproduzcan las acciones del usuario y verifiquen los resultados esperados.

3. Desarrollar un conjunto de utilidades y funciones reutilizables para facilitar la escritura y ejecución de pruebas en Selenium:

    - Identificar patrones comunes en los scripts de prueba y agruparlos en funciones reutilizables.

    - Crear bibliotecas personalizadas que agrupen datos para facilitar la parametrización de las pruebas.

4. Evaluar la eficacia y eficiencia de la automatización de pruebas en comparación con las pruebas manuales:

    - Comparar los resultados y la cobertura de las pruebas automatizadas con las pruebas manuales realizadas anteriormente.

    - Analizar y medir el ahorro de tiempo y recursos obtenido mediante la automatización de pruebas.

    - Evaluar la fiabilidad y la estabilidad de las pruebas automatizadas en comparación con las pruebas manuales.

5. Documentar el proceso de automatización de pruebas y proporcionar pautas para futuros proyectos de automatización:

    - Crear una documentación clara y completa que describa los pasos, las técnicas y las mejores prácticas seguidas durante el desarrollo de la automatización de pruebas.

    - Proporcionar instrucciones detalladas para configurar el entorno de prueba y ejecutar las pruebas automatizadas.

    - Compartir lecciones aprendidas, recomendaciones y consejos útiles para ayudar a otros proyectos futuros de automatización de pruebas basados en Selenium.

Estos objetivos específicos brindan una idea más clara de las tareas y resultados que se esperan de este Trabajo de Fin de Grado. Logrando cumplir con ellos, se pretende contribuir a la mejora de la calidad y eficiencia de las pruebas en aplicaciones web, permitiendo una detección temprana de errores y reduciendo el esfuerzo y tiempo requeridos para las pruebas manuales.

Además, al desarrollar utilidades y funciones reutilizables se fomenta la reutilización del código y la modularidad de las pruebas, lo que facilita el mantenimiento y la escalabilidad de los casos de prueba.

Por último, documentando el proceso de automatización de pruebas, los conocimientos y la experiencia adquirida trabajando en este proyecto servirán como pautas y recomendaciones para futuros proyectos de automatización, fomentando el uso de la automatización de pruebas dentro del proceso de desarrollo de software.

En resumen, estos objetivos específicos permitirán otorgar al proyecto el enfoque de un QA que está realizando tareas de automatización, ya sea por primera vez en su proceso de aprendizaje como para cimentar sus conocimientos y realizar pruebas que aún nunca ha puesto en práctica, aumentando así su valor como tester.

4. Target

Como se ha adelantado, el sitio web que se desarrollará a lo largo de este proyecto estará destinado a un target claro: el personal de QA de las empresas, que se desglosa en dos grandes grupos:

a. Estudiantes en prácticas.

Cuando un estudiante finaliza sus exámenes y entra a realizar las prácticas en una empresa, el testing es uno de los ámbitos en los que pueden querer formarle. Tanto si lo hacen de manera directa (formación personal) como indirecta (formación mediante vídeos y tutoriales), este puede encontrarse con ciertas dificultades a la hora de practicar aquello que le han enseñado.

En la experiencia personal que he vivido al comenzar la Formación en Centros de Trabajo, una situación con la que puede encontrarse un alumno es que se le asigne a un equipo de QA, siendo un ámbito sobre el que no tiene ningún conocimiento, debido a que, generalmente, el ciclo de DAW no incluye ningún tipo de enseñanza al respecto.

Bajo esta premisa, el alumno se encuentra con otra dificultad, y es que, al intentar seguir la formación recibida en el centro de trabajo, la mayoría hacen referencia a vídeos en los que se utilizan bien sitios web ya no disponibles o bien repositorios que o han recibido modificaciones o están obsoletos por utilizar versiones antiguas de software.

Bajo esta premisa se define el primero de los dos grandes grupos que conforman el target del proyecto:

- Estudiantes o personas en prácticas que desean aprender sobre la automatización de pruebas utilizando Selenium, provenientes tanto de carrera universitaria como de módulos de formación profesional.
- Están interesados en adquirir habilidades y conocimientos prácticos en la automatización de pruebas para mejorar su perfil profesional y tener una ventaja competitiva en el mercado laboral.
- Buscan una plataforma o herramienta que les proporcione una experiencia de aprendizaje interactiva y práctica para desarrollar sus habilidades en la automatización de pruebas con Selenium y otras herramientas de testing.

- Necesitan un entorno de prácticas seguro donde puedan experimentar, probar y aplicar los conceptos y técnicas aprendidas en un contexto real sin necesitar interactuar con una página web destinada a otros usos.

b. Formadores.

Del mismo modo, las personas que la empresa destine a formar a la gente de prácticas, o a gente dentro de la empresa que esté recibiendo una formación para cambiar de puesto dentro de la misma, pueden tener dificultades a la hora de encontrar sitios web que sugerir para realizar pruebas sobre la materia impartida.

Con este análisis, se define el segundo gran grupo al que va destinado este Trabajo de Fin de Grado:

- Profesores o profesionales que trabajan en empresas de desarrollo de software y están a cargo de personas en prácticas.
- Buscan una solución integral que les permita proporcionar a sus alumnos un entorno de pruebas práctico y realista utilizando Selenium.
- Necesitan una plataforma que facilite la creación y gestión de pruebas automatizadas, y que sea fácil de usar y entender para sus alumnos.
- Desean brindar a sus alumnos la oportunidad de aplicar los conceptos teóricos aprendidos en clase o en el entorno laboral en un contexto práctico y relevante.
- Buscan una herramienta que proporcione informes y resultados detallados de las pruebas automatizadas para evaluar el rendimiento y el progreso de sus alumnos.

En resumen, el proyecto está dirigido tanto a estudiantes y personas en prácticas que buscan aprender y mejorar sus habilidades en la automatización de pruebas con Selenium, como a profesores y profesionales que desean proporcionar un entorno de prácticas completo y efectivo para las personas que tienen a su cargo. El objetivo es ofrecer una plataforma que cumpla con las necesidades de aprendizaje y formación en la automatización de pruebas, brindando una experiencia práctica y valiosa.

5. Desarrollo.

Por todo lo expuesto, la aplicación consistirá en una interfaz muy básica, de uso sencillo y amigable, puesto que su objetivo, como hemos visto, no consiste en ofrecer un sitio web bonito o con una utilidad práctica para el usuario final, como pueden ser los casos de una tienda online o del dominio de un negocio físico que quiere exponerse en internet para alcanzar un alcance mayor.

Es por esto que se indagará a continuación en profundidad sobre los fundamentos del testing, y de su automatización en contrapartida a las pruebas manuales realizadas típicamente por los desarrolladores.

Además, se mostrarán, con el apoyo de imágenes, partes de la web, explicando su razón de ser y su utilidad.

a. Testing para un estándar de calidad.

En el desarrollo de software, el testing es una disciplina esencial que permite garantizar la calidad y fiabilidad de las aplicaciones, tanto web como de escritorio. A lo largo de la historia, el testing ha evolucionado respondiendo a los avances tecnológicos y la creciente demanda de los usuarios y empresas. Inicialmente, el proceso de testing se basaba en pruebas manuales, donde los testers revisaban y probaban manualmente cada función y aspecto de una aplicación web. Esta metodología era lenta, además de muy propensa a errores humanos, lo que la hacía poco eficiente.

Con el tiempo, se fueron adoptando enfoques más estructurados y metodologías de testing, como las pruebas basadas en casos de uso y las pruebas de regresión. Estas técnicas permitían una cobertura más amplia de los distintos escenarios de prueba, así como una mayor confiabilidad en la detección de errores. Además, se desarrollaron frameworks y herramientas destinadas específicamente al testing en aplicaciones web, lo que facilitó la automatización de ciertas tareas repetitivas y la ejecución de pruebas en entornos con diferentes configuraciones.

El testing ha adquirido una importancia cada vez mayor debido a la creciente complejidad de las aplicaciones web y la necesidad de ofrecer una experiencia de usuario fluida y sin errores. Además, con el auge de la tecnología móvil y la gran variedad de navegadores y dispositivos distintos, el testing se ha convertido en un requisito imprescindible para asegurar la compatibilidad y funcionalidad de las aplicaciones en diferentes plataformas.

Pero el testing no acaba al finalizar el proceso de desarrollo, ya que además de detectar errores y fallos, el testing también contribuye a la mejora continua del producto y la optimización del rendimiento, permitiendo identificar áreas de mejora, evaluar la usabilidad y garantizar la seguridad de las aplicaciones. Por todo esto, el testing es un pilar fundamental en el desarrollo de software para alcanzar estándares de calidad, minimizando los riesgos y ofreciendo productos web en el que los usuarios pueden confiar, y con el que pueden tener la tranquilidad de trabajar en una aplicación libre de fallos.

Sin embargo, con la evolución tanto del número de aplicaciones desarrolladas como de la complejidad de estas, el testing manual dejó de ser suficiente.

b. Automatización.

La automatización de pruebas en el desarrollo de aplicaciones web ha revolucionado la forma en que se realizan las pruebas de software. A medida que las aplicaciones web se volvieron más complejas y el ciclo de desarrollo se aceleró al aumentar las exigencias de los clientes, se hizo evidente la necesidad de adoptar enfoques más eficientes y rápidos para el testeo de los productos desarrollados. Esto llevó al surgimiento de la automatización de pruebas, consistente en utilizar scripts que ejecutan las pruebas de forma automatizada, mediante el uso de software desarrollado para tal fin.

Los primeros frameworks y herramientas para automatizar pruebas surgieron alrededor de la década de los 80, siendo uno de los primeros el framework SQA (Software Quality Assurance), desarrollado por IBM, que se centraba principalmente en la automatización de pruebas a nivel de interfaz de usuario.

Con el avance de la tecnología y la aparición de nuevas herramientas y lenguajes de programación, la automatización de pruebas fue evolucionando. Así, uno de los principales frameworks de automatización, Selenium, surgió en 2004, permitiendo la automatización de pruebas a nivel de navegador, y que cuenta con una gran variedad de lenguajes, como Java, Python y C#, para escribir scripts de pruebas, abriendo así nuevas posibilidades para la automatización de pruebas.

En los últimos años, la automatización de pruebas ha experimentado importantes avances. Se han desarrollado frameworks y herramientas más potentes y sofisticadas, como Jmeter y Postman, que ofrecen capacidades de prueba más amplias, como pruebas de rendimiento, pruebas de seguridad y pruebas de API.

Además, debido a la evolución del paradigma organizativo en las empresas dedicadas al desarrollo de software con la inclusión del método DEV/OPS y la integración continua y entrega continua (CD/CI), la adición de pruebas automatizadas en los procesos de desarrollo se ha vuelto cada vez más común.

Hoy en día, la automatización de pruebas es ampliamente utilizada en el desarrollo de aplicaciones web y móviles. Permite la ejecución rápida de pruebas, y, sobre todo, su fácil repetición, lo que ahorra tiempo y recursos en comparación con las pruebas manuales. La automatización también facilita la detección temprana de errores y la identificación de problemas de manera más eficiente, como ya se ha comentado al hablar de los cinco entornos que se diferencian en el proceso de desarrollo, lo que contribuye a mejorar la calidad del software, reduciendo en gran medida los errores.

Cabe mencionar que, a pesar de los avances, la automatización de pruebas presenta algunos problemas. Por un lado, requiere de habilidades técnicas específicas para desarrollar y mantener los scripts de prueba. Por otro lado, la creación de casos de prueba automatizados puede llevar bastante tiempo y esfuerzo. Además, no todas las pruebas son adecuadas para la automatización, y en algunos casos las pruebas manuales siguen siendo necesarias para ciertos escenarios. De hecho, toda automatización de pruebas requiere de una previa realización de pruebas manuales, ya que debe indagarse sobre las distintas funcionalidades de la aplicación y localizar todos los elementos interactuables de los que cuenta.

c. Interfaz de la aplicación.

La aplicación realizada en este proyecto cuenta con cinco tipos de archivos, uno de los cuales, en lenguaje java, se explicará en el apartado sobre las pruebas automatizadas.

En este sentido, cada página de la aplicación consta de tres archivos: un html, que da forma y estructura a la interfaz del usuario; una hoja de estilos css, que modifica el aspecto visual de los distintos elementos, incluido el cuerpo de la página, y los vuelve responsive, adaptándose a cada tipo de pantalla; y unn con extensión js, donde se incluyen los scripts en lenguaje javascript que producen las distintas acciones que ocurren al interactuar con la aplicación.

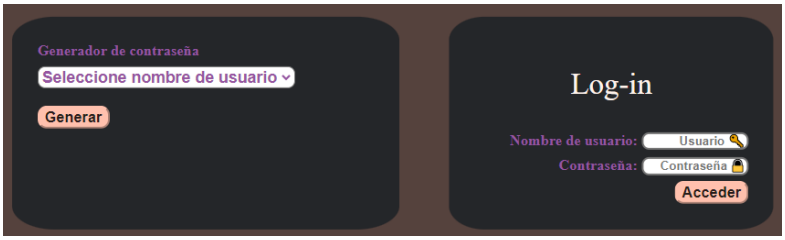
Además de esto, existen varios archivos php, escritos en dicho lenguaje, que permiten interactuar con la base de datos. Esta ha sido creada utilizando la herramienta WAMP, mediante phpMyAdmin.

```sql
CREATE TABLE users (
  ID INT AUTO_INCREMENT PRIMARY KEY,
  USERNAME VARCHAR(20) NOT NULL,
  PASSWORD VARCHAR(15) NOT NULL,
  USER_TYPE ENUM('admin', 'reg_user') NOT NULL DEFAULT 'reg_user'
);
```

La sentencia CREATE que aparece en la imagen es la encargada de crear la tabla que incluirá los datos necesarios para el buen funcionamiento de la aplicación. Esta consta de cuatro campos: id, username, password y user_type: el primero será auto generado, creando un índice único para cada entrada en la tabla; el segundo, username, será el nombre de usuario, que corresponderá a cada una de las páginas de las que consta el proyecto; después, la contraseña asociada al usuario, que le permitirá acceder al apartado correspondiente dentro del sitio web; por último, el tipo de usuario. Se han creado dos: el regular user (reg_user) y el admin. Este último, si bien actualmente no conlleva ninguna responsabilidad, podría llegar a ser útil en algún momento para labores de mantenimiento.

```sql
INSERT INTO users(USERNAME, PASSWORD, USER_TYPE) VALUES
        ('admin', '123456789', 'admin'),
        ('form&radio', 'form1234', 'reg_user'), ('button', 'button1234', 'reg_user'),
        ('dinamicElements', 'dinamic1234', 'reg_user'), ('modalWindow', 'modal1234', 'reg_user'),
        ('alertWindow', 'alert1234', 'reg_user'), ('menu&list', 'menu1234', 'reg_user'),
        ('otherInput', 'input1234', 'reg_user'), ('movingElements', 'moving1234', 'reg_user'),
        ('auxiliarUser1', '2468', 'reg_user'), ('auxiliarUser2', '2648', 'reg_user'),
        ('auxiliarUser3', '2684', 'reg_user'), ('auxiliarUser4', '2846', 'reg_user'),
        ('auxiliarUser5', '2864', 'reg_user'), ('errorUser1', '8642', 'reg_user'),
        ('errorUser2', '8462', 'reg_user'), ('errorUser3', '4862', 'reg_user'),
        ('errorUser4', '6482', 'reg_user'), ('errorUser5', '4682', 'reg_user');
```

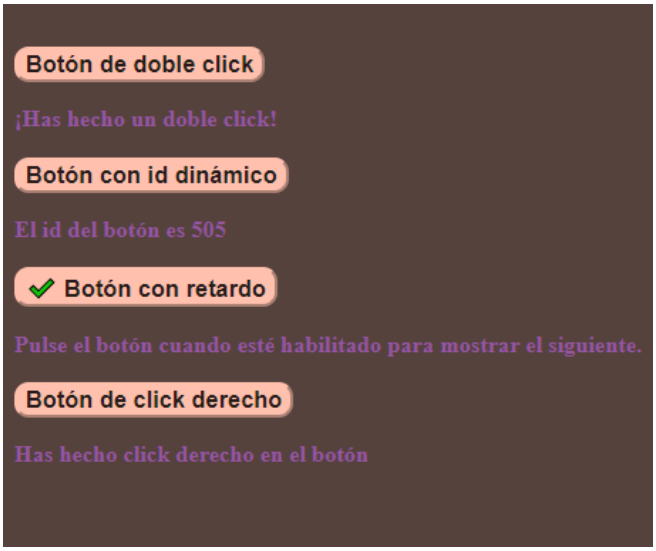
Además, y como puede apreciarse en la imagen superior, se han creado varios usuarios auxiliares y de error para implementar, en un futuro, funcionalidades extra y algunas pantallas de error a las que el usuario deba enfrentarse. Esto, debido al escaso tiempo otorgado para realizar este proyecto, no se implementará en la versión expuesta, si no que formará parte de actualizaciones posteriores.



En esta imagen podemos ver parte de la página de log-in. Como se ha explicado ya, y se aprecia claramente, el fin de este proyecto no es el de ofrecer una interfaz bonita como el que pueden necesitar otras aplicaciones para llamar la atención de un cliente o usuario de algún negocio: aquí, la importancia reside en la variedad de funcionalidades que ofrece

de cara a una automatización de pruebas. Es por ello que, ya en el log-in, vemos una diferencia fundamental con respecto a otras aplicaciones: no es el usuario quien se registra en la página, si no la página la que ofrece unos datos de acceso predefinidos. En el cuadro de la izquierda, el usuario selecciona, en un desplegable, el tipo de funcionalidad que quiere probar. Al pulsar en el botón 'Generar', la aplicación devolverá la información de acceso asociada a esta. Es entonces cuando se deben introducir ambos datos, usuario y contraseña, en el cuadro de la derecha para acceder a la página escogida.

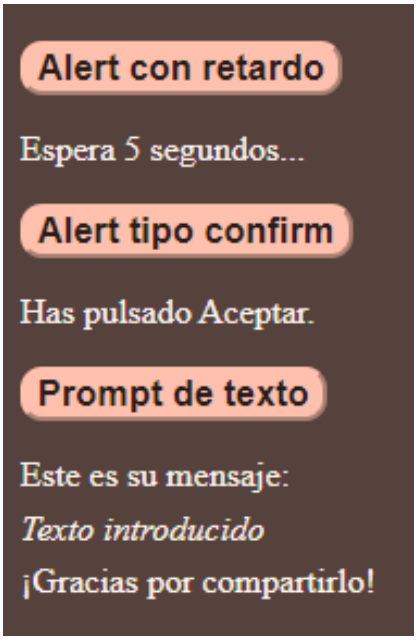
Esta forma de proceder, contraintuitiva para el usuario estandar, es lo que proporciona un mejor entorno de pruebas al estudiante en prácticas de automatización de testing. Una vez introducidos los datos correspondientes, se accederá a la página asociada.



En este caso, esta página muestra unos botones con distintas funcionalidades: uno de ellos requiere de un doble click para que su función haga efecto; otro, contiene un id dinámico, adquiriendo un valor numérico comprendido entre el 0 y el 1000 cada vez que se recargue la página; el siguiente, contiene una espera de cinco segundos, antes de los cuales estará deshabilitado; tras esperar este tiempo y pulsar en él, aparecerá el cuarto y último botón, con el cual sólo se podrá interactuar mediante un click derecho.

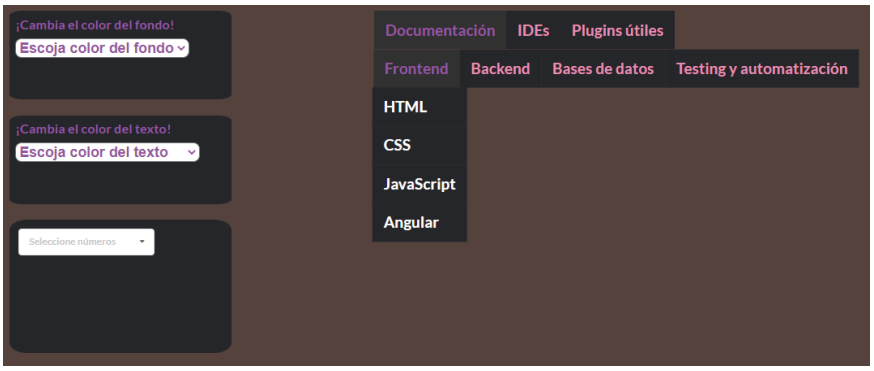
Todos ellos ofrecen un mensaje tras ser ejecutados. esto es un detalle importante, puesto que la automatización de un sitio web debe incluir la comprobación de mensajes que aparecen en pantalla, tanto de manera estática como aquellos que surgen por la interacción del usuario. El código html de la siguiente imágen deja patente que el aspecto visual que se ofrece al usuario es muy simple, siendo el punto menos importante de este tipo de páginas web:

```
<div>
    <button ondblclick="doubleClick()">Botón de doble click</button>
    <p class="pButton" id="clickDoble"></p>

    <div id="contenedor">
    </div>
    <p class="pButton" id="clickDinamico"></p>

    <button id="delayedButton" onclick="botonInvisible()">&#x2716;&#xfe0f; Botón con retardo</button>
    <p class="pButton">Pulse el botón cuando esté habilitado para mostrar el siguiente.</p>

    <button oncontextmenu="rightClick()" id="rightButton" hidden="true">Botón de click derecho</button>
    <p class="pButton" id="clickDerecho"></p>

</div>
```



Este es un fragmento de otra de las páginas a las que puede acceder el usuario. En ella, se ofrecen varios tipos de mensajes emergentes mediante el uso de alerts y otros comandos javascript. Es importante que un trabajador QA sepa interactuar con este tipo de elementos, puesto que, normalmente, estos impiden la interacción con el resto de la página, lo que, unido al uso tan extendido de estos, hacen que sea imprescindible saber cómo tratar con ellos.

En este caso, se han utilizado tres tipos de alert: uno, el más normal, cuenta con un retardo de 5 segundos, tras los cuales ofrecerá una ventana emergente con un mensaje de aviso y un único botón de 'aceptar'; el segundo, similar pero con dos opciones: 'aceptar' y 'cancelar'. Cada una de ellas realizará una acción diferente, que en este caso será mostrar un mensaje diferente que, como ya se ha comentado, será importante para la comprobación que debe realizar quien esté automatizando la página; por último, se ofrece un prompt con un mensaje de información y un espacio para escribir, el cual cuenta con un mensaje pre establecido (placeholder). Este también realizará acciones diferentes si se escribe algo o si no, y si se pulsa en 'aceptar' o en 'cancelar'.



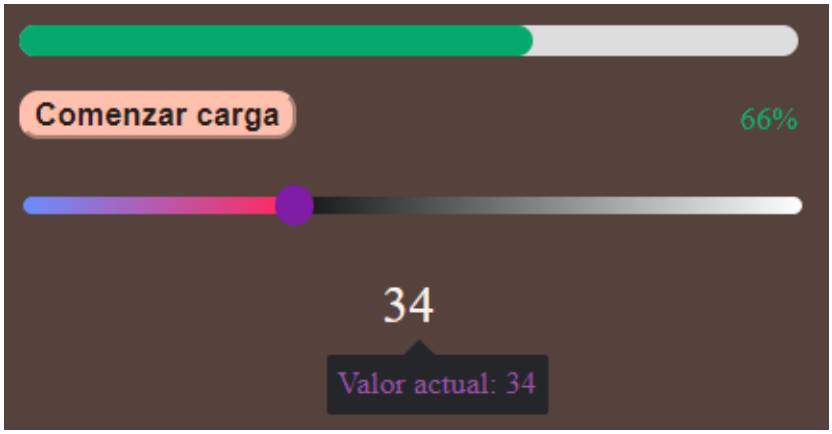
En la imagen que se muestra junto a este texto, podemos ver cómo se ofrecen dos tipos de elementos: en la parte superior, nos encontramos con una lista de elementos tomando la forma de un menú desplegable, el cual cuenta con varios enlaces mostrados en diferentes estratos. Esto permite poner a prueba, en la automatización, la interacción que implica situar el cursor del ratón sobre un elemento, aunque no se haga click en él. De esta manera, se fuerza a la persona en proceso de formación de QA a poner a prueba otro tipo de interacciones relacionadas con el uso automatizado del ratón.

Por otra parte, a la izquierda, tenemos varios desplegables que nos permitirán escoger, en orden de arriba a abajo, el color de fondo del contenedor del desplegable, el color del texto que acompaña al desplegable y los distintos números que el usuario quiera mantener como valores escogidos del desplegable, respectivamente. Esto ofrece la posibilidad al usuario de comprobar no solo los elementos visibles en la pantalla, si no también los valores de los atributos que contienen estos elementos: podrá realizar una prueba que se asegure que, al seleccionar el color verde como fondo del contenedor del primer desplegbale, dicho contenedor adquiere, efectivamente, el valor 'green' para su atributo background-color. Esto se lleva a cabo mediante un código javascript que interfiere en los valores que la hoja de estilos css proporciona a dicho elemento, como puede apreciarse en las imágenes.

```javascript
const colorSelect = document.getElementById("colorSelect");
const bgColor = document.getElementById("bgColor");

colorSelect.addEventListener("change", function() {
    const selectedColor = colorSelect.value;
    bgColor.style.backgroundColor = selectedColor;
});

const textColorSelect = document.getElementById("textColorSelect");
const textColor = document.getElementById("textColor");

textColorSelect.addEventListener("change", function() {
    const selectedColor = textColorSelect.value;
    textColor.style.color = selectedColor;
});
```

```html
<div id="bgColor" class="leftElements topElements divSmall width20">
    <label for="">¡Cambia el color del fondo!</label>
    <span class="brSmall"></span>
    <select name="" id="colorSelect">
        <option value="" hidden>Escoja color del fondo</option>
        <option value="green">Verde</option>
        <option value="red">Rojo</option>
        <option value="white">Blanco</option>
        <option value="yellow">Amarillo</option>
    </select>
</div>

<span class="brMedium"></span>

<div class="leftElements divSmall width20">
    <label id="textColor">¡Cambia el color del texto!</label>
    <span class="brSmall"></span>
    <select name="" id="textColorSelect">
        <optgroup label="Colores primarios">
            <option value="" hidden>Escoja color del texto</option>
            <option value="blue">Azul</option>
```

La principal particularidad de esta página es la del manejo de distintas ventanas del navegador. Desde este apartado, el usuario podrá abrir tanto una nueva pestaña como una nueva ventana, así como una ventana de navegador que muestra un texto predefinido. Esto obliga a que, en la automatización, utilice ciertos comandos destinados al manejo del navegador que se verán más adelante, en el apartado e de este punto, 'Pruebas automatizadas'.



Por último, merece la pena mencionar este apartado de la aplicación, puesto que, a diferencia del resto, muestra elementos cuyo contenido varía con el tiempo, sin depender de la interacción del usuario. También podemos ver una diferencia reseñable en la hoja de estilos asociada a esta página:

```
177  p#tooltip-value {
178      display: none;
179      position: absolute;
180      top: 2.2vw;
181      left: -3.5vw;
182      background: #242629;
183      padding: 0.4vw;
184      font-size: 1.2vw;
185      color: #9656a1;
186      border-radius: 2px;
187      animation: fadeIn 2.5s;
188  }
189  @keyframes fadeIn {
190      from {
191          opacity: 0;
192      }
193      to {
194          opacity: 1;
195      }
196  }
```

Como se puede apreciar, se emplea el uso del keyframes. Esta regla css permite controlar los pasos que seguirá una animación que queramos declarar sobre un elemento. En este caso, se modifica la animación fadeIn, que permite aplicar un estilo de fundido a un elemento al aparecer. Con el parámetro opacity, se define la opacidad que ese elemento tendrá al comenzar su carga (from) y la que tendrá al terminar la misma (to). Además, en la definición de la animación se incluye el tiempo que esta carga tomará, en este caso 2 segundos y medio, haciendo así que el efecto de fundido sea más personalizable, obteniendo el efecto deseado.

d. Uso de la aplicación.

Vista la interfaz, y algunas partes del código, de la aplicación, el siguiente punto es hablar de su funcionalidad. Como se ha adelantado en varios puntos de este documento, la aplicación está orientada a servir como base para la realización de pruebas de testing automatizadas. Para tal fin, y como se vió en el apartado anterior, se ponen a disposición un gran número de elementos interactuables de tipos muy diversos que permiten poner en práctica cualquier tipo de interacción.

La forma en la que un usuario, recordemos, estudiante o persona en prácticas para desarrollarse como QA, interactuará con esta aplicación es la siguiente:

- En primer lugar, y como primer paso fundamental en cualquier labor de automatización de pruebas, el usuario deberá familiarizarse manualmente con el sitio web: accederá a la página de log-in y, siempre con las opciones de desarrollador habilitadas en el navegador, revisará uno por uno todos los elementos, interactuables y no, que estén a la vista. Hará click en cualquier elemento que lo permita, como el desplegable que se vió en el apartado anterior de la sección de log-in, probará a escoger cada una de las opciones disponibles viendo el resultado que arroja el botón de 'generar' con cada una de ellas, y comprobará qué sucede si intenta acceder sin completar los campos requeridos, o introduciendo unos erróneos. Tras esto, introducirá unos datos válidos, y repetirá el proceso de descubrimiento de la página en cada una de las opciones que la aplicación proporciona.

- Después, creará una hoja de ruta para la automatización: tratará de analizar la mejor manera de aproximarse a una automatización que permita probar todo lo que la aplicación ofrece. En este sentido, resulta muy útil el botón de 'Logout' habilitado en la esquina superior derecha de cada página, ya que le permite que, tras realizar cada prueba, pueda volver a la página de inicio para volver a iniciar el proceso con la siguiente página, ingresando las credenciales correspondientes.

- Como parte final, que se desarrollará en el siguiente punto, el usuario creará los test que considere oportunos. No hay una sola manera de aproximarse a la automatización de un sitio web, al igual que no la hay cuando el usuario de una tienda online accede para realizar sus compras: hay personas que prefieren buscar en la sección del tipo de ropa que más les gusta, por ejemplo, camisetas, y después pasar a los que menos interés le generan, mientras que otras prefieren ordenar todo el catálogo de la tienda por orden de precios e ir descubriendo, en cualquier categoría, las gangas que le puedan resultar interesantes. Del mismo modo, aunque el paso previo inevitable será acceder a través del log-in, habrá quienes prefieran explorar primero cada página de elementos interactuables, accediendo con las credenciales nada más abrir la página de log-in, y dejar la exploración de la propia página de log-in para el final, y habrá quienes lo hagan a la inversa.

e. Pruebas automatizadas.

Ya se ha mencionado cómo, en este proyecto, se utilizan cinco tecnologías. En los anteriores apartados se han visto ya cuatro de ellas: html, css, javascript y php. La última, java, es la que se utiliza para realizar la automatización de la aplicación.

Si bien Selenium permite realizar pruebas en múltiples lenguajes, permite C#, Ruby, php, Python, Perl y Groovy, se ha decidido utilizar Java en este proyecto por lo versátil que resulta, las grandes posibilidades que ofrece y lo ampliamente utilizado que es. Sin embargo, es un punto muy positivo de este proyecto el que puedan poner en práctica los conocimientos de Selenium gente acostumbrada a lenguajes muy diversos, ya que permite conocer una nueva tecnología sin tener que aprender también un nuevo lenguaje de programación.

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>selenium_tfg</groupId>
    <artifactId>selenium_tfg</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>4.8.3</version>
        </dependency>
        <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>7.7.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

Selenium utiliza frameworks de testing para funcionar, por lo que, a la hora de crear un proyecto de automatización, debemos tener esto en cuenta: en primer lugar, el usuario debe tener instalado Maven en su sistema, ya que un proyecto Java para la utilización de Selenium debe ser de tipo Maven. Además, deberá añadir las dependencias correspondientes en el archivo pom.xml, generado automáticamente al crear un proyecto Maven, para que tanto Selenium como el framework de testing escogido puedan funcionar correctamente. En este caso, se ha escogido TestNG, pues proporciona funcionalidades fáciles de implementar sin dejar de ser por ello muy completo. jUnit es otra buena opción como framework de testing para trabajar con Selenium.

Una vez implementadas las dependencias en el pom del proyecto, se pueden empezar a desarrollar los archivos, .java en este caso, que realizarán las pruebas.

Al tratarse la automatización, como ya se ha dicho, de un campo en gran parte desconocido, es importante dar unas pautas generales de la manera de trabajar para desarrollar las pruebas. Y la principal, e imprescindible, es la ya mencionada en el apartado anterior, al hablar de cómo el QA debe realizar primero un acercamiento manual a la aplicación: trabajar siempre con las opciones de desarrollador del navegador abiertas.

La automatización requiere de forma recurrente, prácticamente en cada paso que se da, de localizar los elementos existentes a la vista del usuario. Esto se hace mediante el uso de selectores, que permiten apuntar hacia los elementos recurriendo a xpath, a css o al id, si el elemento lo tuviese. Un ejemplo sería el siguiente:

```
116        chrome.findElement(By.xpath(xpathExpression:"(//p[@class='pButton'])[1]"));
```

En esta imagen puede apreciarse cómo se utiliza xpath para localizar un elemento por medio de su clase. Si bien una clase puede afectar a varios elementos, y por tanto no ser un localizador preciso, se pueden combinar varios selectores para realizar una búsqueda más precisa. El mejor

atributo para realizar la búsqueda de un elemento es siempre el id, puesto que este debe ser único en cada página. Sin embargo, muchas páginas no se molestan en indicar id en todos los elementos, lo cual es una mala práctica del diseño web. Además, en ocasiones, los id son dinámicos, lo que imposibilita este método de localización. Un ejemplo de esto se ha visto anteriormente en la documentación al mostrar la página button.

```
101     WebElement passInput = chrome.findElement(By.name(name:"password"));
102     passInput.sendKeys(password);
103     chrome.findElement(By.name(name:"btn-login")).click();
```

Una vez localizado el elemento deseado, se debe realizar la interacción que se busca tener con el mismo. Esto puede ir desde hacer click, en el caso de un button, hasta rellenar un campo de texto con una cadena de caracteres concreta mediante el comando sendKeys. Ambos ejemplos se pueden apreciar en la imagen superior. Con mayor o menor dificultad, podemos realizar cualquier tipo de acción que sea posible realizar de forma manual. Esto quiere decir que no se puede, por ejemplo, escribir texto en un elemento button, aunque sí se puede hacer click en cualquier parte de la pantalla por medio de coordenadas, lo cual puede resultar muy útil en algunas circunstancias.

Además de estas acciones, muy socorridas, también existen otras que son menos visuales, pero muy importante a la hora de realizar pruebas en una aplicación. Son las que implican corroborar la información mostrada en la página: mensajes de error, información mostrada en la interfaz, campos de texto rellenados por el usuario, etcétera. Estas comprobaciones se realizan mediante el uso de aserciones, comandos que cotejan datos entre sí. Suelen acompañarse de los comandos getText y contains, que, usados en los elementos localizados, permiten recoger la información contenida en ellos.

```
107     String textoDinamico = chrome.findElement(By.id(id:"clickDinamico")).getText();
108     String buttonId = chrome.findElement(By.xpath(xpathExpression:"//button[text()='Botón con id dinámico']"))
109     .getAttribute(name:"id");
110     Assert.assertTrue(textoDinamico.contains(buttonId), buttonId);
```

Con estas cuatro acciones básicas, y aplicando tanto la correcta localización de elementos como la lógica a la hora de realizar la navegación por la aplicación a través de la automatización, puede realizarse un set de pruebas robusto que permita comprobar las funcionalidades de todo el sitio web.

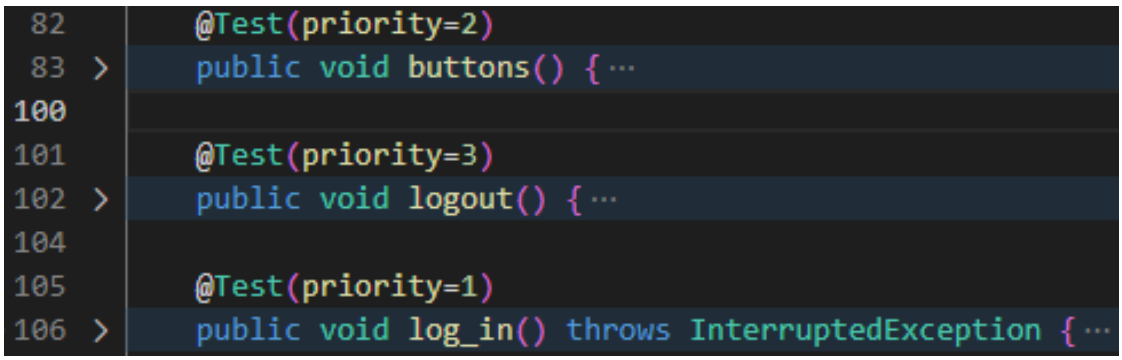
En cuanto a la estructura, es importante también mencionar la manera en que se ejecutan los test. Mediante el uso de los framework ya mencionados, jUnit y TestNG, se deben crear, en el archivo .java, diferentes métodos precedidos de la notación @Test. Cada uno de estos métodos será una prueba independiente, pero hay que tener en cuenta que seguirán un orden secuencial: esto quiere decir que el segundo test que se ejecute lo hará partiendo desde el punto en el que el primero ha terminado, lo que puede crear errores de ejecución: si, por ejemplo, el primer test termina tras comprobar un elemento de la página button.php y el segundo empieza intentando realizar log-in en la página correspondiente, este fallará, puesto que no podrá encontrar ni el input en el que debe escribir las credenciales de acceso ni pulsar el botón para tal fin.

En este contexto es en el que entran en juego las notaciones @Before y @After. Estas otorgan la posibilidad de indicar una serie de acciones que se realizarán antes o después, bien de cada test, bien de todos los test, o bien ambas, modificando ligeramente la notación. Estas serían las distintas maneras de aplicarlo en TestNG (en jUnit, si bien existen las mismas variantes, las notaciones varian ligeramente):

- @BeforeMethod. Este bloque de acciones se ejecutará antes de cada @Test definido en el archivo .java.
- @AfterMethod. Este bloque de acciones se ejecutará después de cada @Test definido en el archivo .java.
- @BeforeClass. Este bloque de acciones se ejecutará solamente antes del primero de los @Test definidos en la clase sobre la que se está trabajando.
- @AfterClass. Este bloque de acciones se ejecutará solamente después del último de los @Test definidos en la clase sobre la que se está trabajando.
- @BeforeSuite. Este bloque de acciones se ejecutará antes de cualquier configuración existente para una suite de pruebas completa.
- @AfterSuite. Este bloque de acciones se ejecutará después de finalizar una suite de pruebas completa, generalmente, para desconectarse de una base de datos o tareas similares.

```
72  @BeforeClass
73  public void setUp() {
74
75      System.setProperty(key:"webdriver.chrome.driver", value:".\\src\\test\\resources\\chromedriver.exe");
76      chrome = new ChromeDriver();
77      chrome.manage().window().maximize();
78      chrome.get(url:"http://localhost/proyecto_tfg/html_files/log_in.html");
79  }
80  @AfterClass
81  public void tearDown() {
82      chrome.quit();
83  }
```

Hay que tener en cuenta que el orden de ejecución de los test puede ser, en ocasiones, azaroso: pueden no siempre seguir el mismo orden, lo que provocaría fallos. Si bien esto se solventa en muchas ocasiones con el uso de las notaciones before y after mencionadas, habrá casos en los que no pueda corregirse con esto. Para ello, existen maneras de ordenar los test para que sigan siempre el orden deseado. De nuevo, la manera de lograr esto será diferente si utilizamos jUnit o TestNG, e incluso puede diferir entre distintas versiones de un mismo framework. Como ejemplo, se muestra a continuación la manera de ordenar los test en TestNG, mediante la notación priority:



Evidentemente, y como ocurre en todos los ámbitos de la informática, las posibilidades que ofrece Selenium en automatización de pruebas es mucho más extensa, pero con la información vista en este apartado es suficiente tanto para entender cómo se trabaja con esta tecnología como para realizar un gran número de test de distintos tipos. Por supuesto, siendo una tecnología que se basa en lenguajes de programación, todo lo que ofrecen estos puede utilizarse en los test, como los arrays, los bucles o los condicionales.

f. Fases del proyecto.

Finalmente, y explicados ya los motivos de la realización de este proyecto, sus objetivos y público al que va dirijido, la interfaz y su funcionalidad, y la manera en que el usuario final interactuará con la aplicación, solo queda explicar el proceso de desarrollo que se ha seguido para lograr el resultado deseado.

Para elaborar esta aplicación, se decidió comenzar por diseñar la estructura de la misma, teniendo en cuenta el número de páginas que debían estar disponibles y los distintos elementos con los que tendría que interactuar el usuario. Con esto en mente, elaborar el front-end de la página de log-in fue la principal prioridad, ya que, al mostrar esta los accesos a las distintas partes de la aplicación, constituía un importante nexo de unión desde el que ramificar el proyecto.

Tras diseñar esta primera página, y viendo que era posible reutilizar gran parte de la hoja de estilos para desarrollar el resto de vistas de la aplicación, se siguió trabajando esa línea, creando una plantilla html y otra css con los elementos comunes a todas las páginas. Esto permitió agilizar el trabajo a la hora de implementar los distintos elementos.

Después, el desarrollo continuó con la base de datos: al ser muy sencilla y pequeña, resultaba oportuno dejar esa parte del proyecto finalizada para poder centrarse en el resto de elementos. Si bien la implementación de la misma dentro de la lógica del código dio más problemas de lo esperado, finalmente, y siempre utilizando lenguaje php, se consiguió implementar correctamente, permitiendo realizar log-in con los distintos usuarios y siendo redireccionado a la página correspondiente.

Una vez que la estructura y las plantillas estuvieron listas y con la implementación de la base de datos, se pasó a crear los elementos con los que se interactuaría en las distintas páginas. Algunos, como los botones o formularios, no supusieron ningún desafío por lo sencillo que resulta incluirlos; otros, como la barra de carga, mostrada anteriormente, o algunos elementos que permiten ser arrastrados a distintos puntos de la página acarrearon mayor complicación.

A la par que se creaban estos elementos html, fue necesario ir generando código javascript para las distintas acciones: mover un elemento, pulsar en un botón, colocar el ratón en un lugar concreto, etcétera. Esto, si bien en algunos contados casos precisó de una investigación más exhaustiva sobre el modo de implementar las acciones deseadas, no retrasó demasiado el proyecto.

Por último, se ha decidido emplear la última semana de la temporalización del proyecto en realizar una automatización completa de la página. Si bien esto no forma parte como tal del proyecto desarrollado, es importante tanto para mostrar la manera en la que el usuario interactuará con la página, ya que, como se ha mencionado anteriormente, la automatización es un campo que en ocasiones cuesta entender si nunca se ha trabajado en él, como para comprobar que, efectivamente, todo lo que quería lograrse con el proyecto se ha conseguido.

Es cierto que la aplicación no cuenta con el acabado que se buscaba al inicio del mismo, pero se trabajará en él en el futuro para acabar implementando todo aquello que queda pendiente en el momento de la entrega de este proyecto. El escaso tiempo de un mes otorgado para la realización de un proyecto desde cero ha hecho que sea inviable finalizarlo completamente.

6. Conclusiones.

En conclusión, el proyecto web desarrollado tiene como objetivo general crear una plataforma que permita a los usuarios practicar y mejorar sus habilidades en la automatización de pruebas, ofreciendo un entorno interactivo y educativo seguro y completo. Los objetivos específicos del proyecto se centran en proporcionar a los alumnos de centros de enseñanza y personas en prácticas en empresas una herramienta accesible y fácil de usar, así como ofrecer a los profesores y supervisores de empleados en prácticas en empresas dedicadas al desarrollo de software un recurso que ofrecer a sus alumnos consistente en un entorno para realizar pruebas con herramientas de automatización como Selenium.

El proyecto está dirigido principalmente a estos dos grupos: los alumnos y las personas en prácticas que desean aprender sobre la automatización de pruebas, y los instructores de estos. Al ofrecer un entorno interactivo y práctico, los usuarios podrán completar sus conocimientos teóricos con habilidades prácticas en la automatización de pruebas, lo que les permitirá destacar en este campo al recibir una formación especializada.

El testing en el desarrollo de software ha demostrado ser crucial para garantizar altos estándares de calidad en las aplicaciones, y ha evolucionado desde pruebas manuales hasta la automatización, permitiendo una mayor eficiencia y precisión en la detección de errores. La automatización de pruebas ha facilitado la rápida ejecución de pruebas repetitivas, ahorrando tiempo y recursos. Además, ha permitido ampliar el alcance de las pruebas, añadiendo a las pruebas funcionales las de de rendimiento, las pruebas de seguridad y las pruebas realizadas sobre APIs.

En el desarrollo de aplicaciones web, la automatización ha experimentado una evolución constante, encontrándose en la actualidad en un estado muy desarrollado. Los profesionales de QA tienen a su disposición herramientas y framewors muy potentes, que les brindan una amplia gama de capacidades para realizar pruebas exhaustivas y eficientes. La plataforma desarrollada en este proyecto busca aprovechar estos avances y brindar a los usuarios una experiencia educativa completa.

En resumen, el proyecto de desarrollo web propuesto ofrece una solución innovadora para mejorar las habilidades de los usuarios en la automatización de pruebas, al mismo tiempo que proporciona una herramienta de enseñanza para los profesores y trabajadores en el campo de la calidad del software. A través de una plataforma interactiva y accesible, los usuarios podrán poner en práctica sus conocimientos teóricos, practicando sus habilidades en un entorno seguro.

7. Bibliografía.

phpMyAdmin. https://docs.phpmyadmin.net/es/latest/

Manual de PHP. https://www.php.net/manual/es/index.php

Lenguaje CSS. https://lenguajecss.com/

Lenguaje HTML. https://lenguajehtml.com/

Lenguaje JS. https://lenguajejs.com/