Memoria Testing Básico.

11/06/2021

—

David Cebrián Cortés, 2º DAW
IES Jacarandá
Brenes, (Sevilla)

Git: https://github.com/davidcebrian/testingBasico

ÍNDICE

1. Presentación.

1. Introducción.

Este proyecto tratará de aprender algo que hemos ido oyendo durante todo el curso pero no se ha visto específicamente, por lo que es interesante hacer un proyecto dedicado a ello y así tener cubiertos más conocimientos necesarios en el ámbito de la programación.

El tema a tratar sería el testing funcional mediante procesos automatizados con Selenium, el cual nos permite hacer pruebas del funcionamiento de la aplicación, grabarlas y automatizarlas y pruebas de testing en Angular mediante Jasmine, la herramienta que permite crear test tanto unitarios, como de integración en Angular; y Karma que es la herramienta de ejecución de esos tests.

También porque es algo de suma importancia en cualquier desarrollo de aplicaciones ya que abarata el costo de producción a largo plazo al prevenir los errores y detectarlos a tiempo para que puedan ser solucionados y no hay mucha gente que se especialice en testing, sobre todo de la parte de angular.

2. Objetivo.

El objetivo es investigar sobre testing, realizando una clasificación de tipos y transmitir ese conocimiento en el proyecto, y algunas herramientas que se utilizan para ello, describiendolo todo brevemente e ilustrando las explicaciones con demos de pruebas realizadas sobre aplicaciones en producción. Empezando por Selenium IDE ya que es más fácil de entender y más intuitivo para pasar a Selenium WebDriver ya que nos permite desarrollar pruebas más complejas.

Y de igual forma realizar una serie de test en Angular mediante Jasmine y Karma tras desarrollar su teoría, también añadiendo demos de esos test para así obtener conocimientos básicos sobre ellos.

Toda la información reunida será después expuesta en mi GitHub con el objetivo de dar apoyo a la gente que como yo quiera informarse sobre testing y no encuentre lo que busca en un sitio concreto.

3. Tecnologías y herramientas.

Tecnologías e IDEs a utilizar	Java 11+, Eclipse, Angular, Visual Studio Code
Sistema control de versiones (Politica)	Git junto a GitHub.
Herramientas de testing	Selenium (Selenium IDE, WebDriver) , Jasmine y Karma.
Herramientas gestión del proyecto (SCRUM)	Pivotal Tracker

Como lenguaje se utilizará **Java en Eclipse** para las pruebas con **Selenium**, así como **Selenium IDE** para realizar unas primeras pruebas automatizadas generando scripts, ya que es una manera sencilla de entender lo que son estas pruebas y como se realizan y **Selenium Webdriver** que es la herramienta que nos permite desarrollar esas pruebas con Java para tener mayor control sobre ellas y darle más complejidad.

Para la realización de las pruebas en angular utilizaremos lenguaje **Typescript** que es el utilizado en Angular y **Jasmine y Karma** para la realización de las pruebas unitarias y de integración en angular, ya que son los que utiliza angular por defecto y nos permite realizar las pruebas necesarias en nuestro código.

Se utilizará la metodología de trabajo **SCRUM** por la agilidad y la capacidad de adaptación y mejora que nos aporta, además de poder ir pequeños objetivos poco a poco.

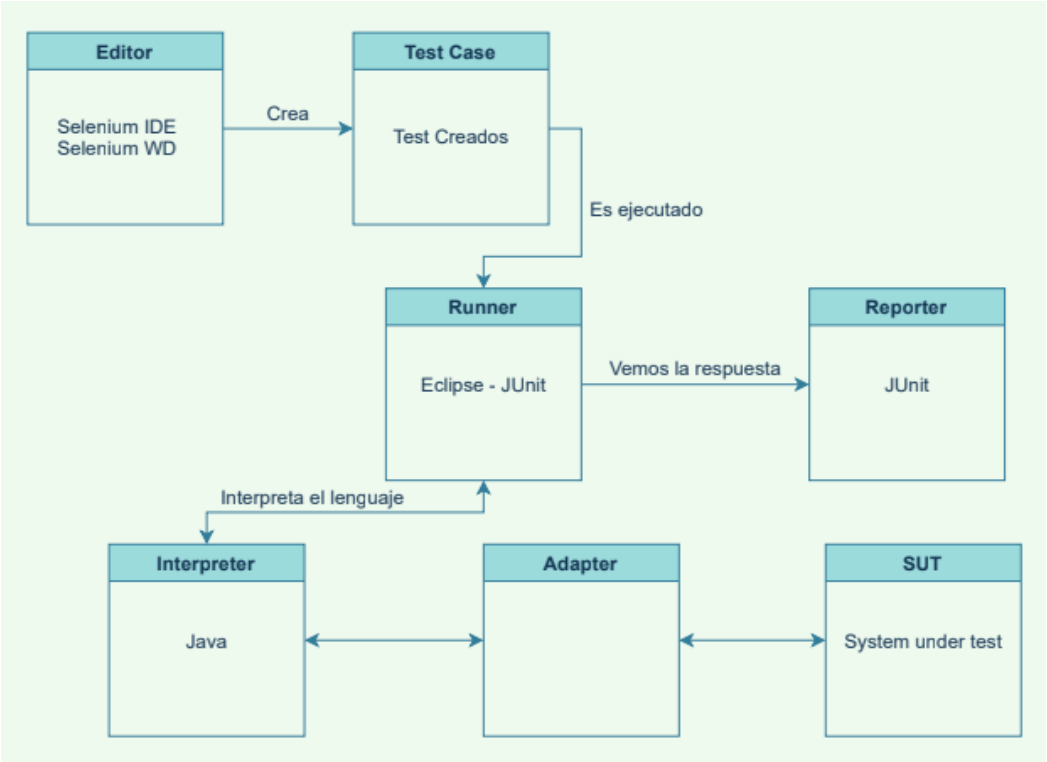
Como herramienta de control de versiones tenemos **GIT** junto con **GitHub,** lo que nos permite llevar un control de versiones exhaustivo de nuestro código y nos da la posibilidad de crear ramas distintas de desarrollo en caso necesario.

Para la gestión del proyecto la herramienta a utilizar será **Pivotal Tracker** porque nos permite describir historias de usuario con mucha precisión y poder subir partes de código o archivos asociados a las propias tareas para llevar un buen control de las tareas que vamos realizando. Y encaja perfectamente con la metodología de trabajo escogida (SCRUM).

2. Diseño del Sistema.

1. Arquitectura del sistema.

La arquitectura del sistema de un sistema de pruebas puede tener una forma como la siguiente:



Donde el **Editor** es cómo y con qué creo y edito mis casos de prueba, como puede ser Selenium IDE, para WebDrive eclipse si utilizo Java, etc.

Test Case (TC) son los casos de pruebas que creamos utilizando un editor específico y un lenguaje para ello, WebDriver por ejemplo utiliza java en Eclipse y necesita un **runner** para ejecutar los test.

Runner es el que ejecuta los tests, puede hacerlo de forma conjunta ejecutando todos los test, solo un grupo de esos test o uno solo y además nos devuelve si estos test han sido exitosos o no.

Interpreter es el encargado de interpretar el lenguaje en el que escribimos nuestros TC. Con el ejemplo de WebDriver tenemos que el interpreter sería el compilador del lenguaje en el que está desarrollado y la biblioteca de Selenium que ayuda a interpretar el código.

El **adapter** son las capas de adaptación entre los TC interpretados y el sistema bajo prueba (System Under Test o SUT). Un ejemplo sería WebDriver usado para automatizar la aplicación a través de la interfaz web.

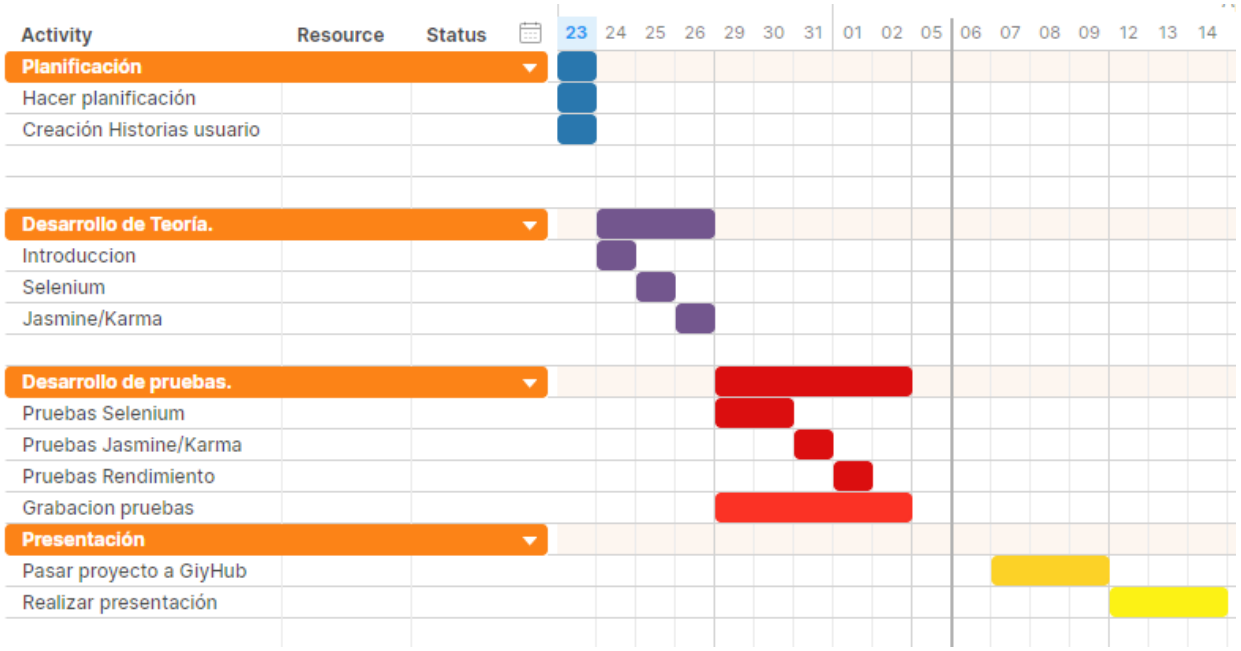
2. Requisitos del sistema.

Los requisitos mínimos que tendrá que tener nuestro sistema para el desarrollo y ejecución de los casos de pruebas que se van a realizar son:

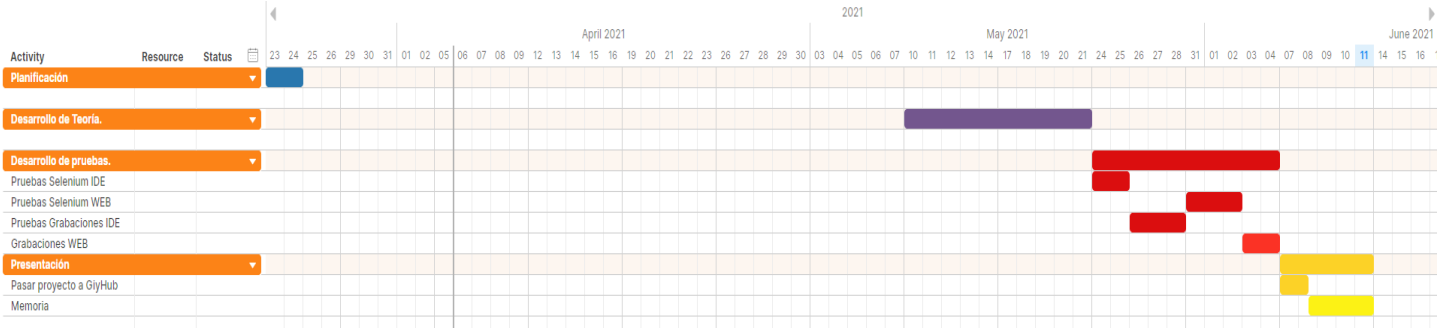
- Java 11 o superior.
- Eclipse.
- Selenium IDE y Selenium WebDriver.
- Driver selenium para el navegador que vayamos a utilizar.
- Angular 10 o superior - Angular CLI
- Navegador. (Google chrome en nuestro caso)
- Streamlabs OBS.
- Visual Studio
- OpenShot

3. Planificación temporal.

Estimación de 3-4 horas diarias:



Dedicación real 2-3h diarias en los dias marcados:



3. Desarrollo del sistema.

1. Planificación de las actividades de desarrollo.

Como actividades de desarrollo tenemos el desarrollo completo de la teoría en cuanto a testing, el cual será dividido en distintas partes notorias y subdivididos en más apartados; y otra parte del desarrollo será la realización de pruebas de testing con Selenium IDE y Selenium Web Driver para apoyar la teoría desarrollada, esas pruebas se grabarán en vídeos explicativos para guiar un poco a todo el que lo vea y quiera empezar a usar estas herramientas.
Las partes más importantes del desarrollo teórico son :

- Índice
- Introducción
- Clasificación de las pruebas
- Herramientas
- Selenium
- Jasmine & Karma

Índice: Pues aquí introduciremos todos los enunciados sobre los que hablaremos, el índice quedará de la siguiente forma:

ÍNDICE
INTRODUCCIÓN.
¿Qué es testing?
CLASIFICACIÓN DE LAS PRUEBAS.
P. Manuales.
P. Automatizadas.
P. Unitarias
Pruebas de integración
P. funcionales

Introducción: Se hará una pequeña introducción al trabajo explicando el porqué de ese desarrollo y que es lo que se hará.

Clasificación de las pruebas: Realizaremos una clasificación somera de los distintos tipos de pruebas que existen dentro del mundo del desarrollo, comparando distintos blogs y distintas clasificaciones para hacer una propia.

Herramientas: Se mencionan algunas herramientas que se utilizan en el mercado tanto de uso gratuito como de pago.

Selenium: Hablaremos sobre la herramienta selenium y las distintas divisiones que tiene, realizando las pruebas pertinentes para apoyar la teoría.

Jasmine & Karma: Se hablará sobre las herramientas de testing Jasmine y Karma ya que son las utilizadas en Angular, una herramienta muy utilizada hoy en día para el desarrollo frontend.

En cuanto al desarrollo de las pruebas con Selenium se realizarán pruebas sobre la página web de PCCOMPONENTES (https://www.pccomponentes.com/). Serán pruebas básicas tales como:

Login en la aplicación.

Busqueda de algun articulo.

Añadir artículo al carro de la compra.

Comprobar que podemos comprar artículos.

Las grabaciones de las pruebas se harán con Streamlabs OBS (https://streamlabs.com/?l=es-ES) y los vídeos serán editados con OpenShot (https://www.openshot.org/es/download/) para posteriormente ser subidos a YouTube (https://www.youtube.com/) e incluir los links en el documento del trabajo.

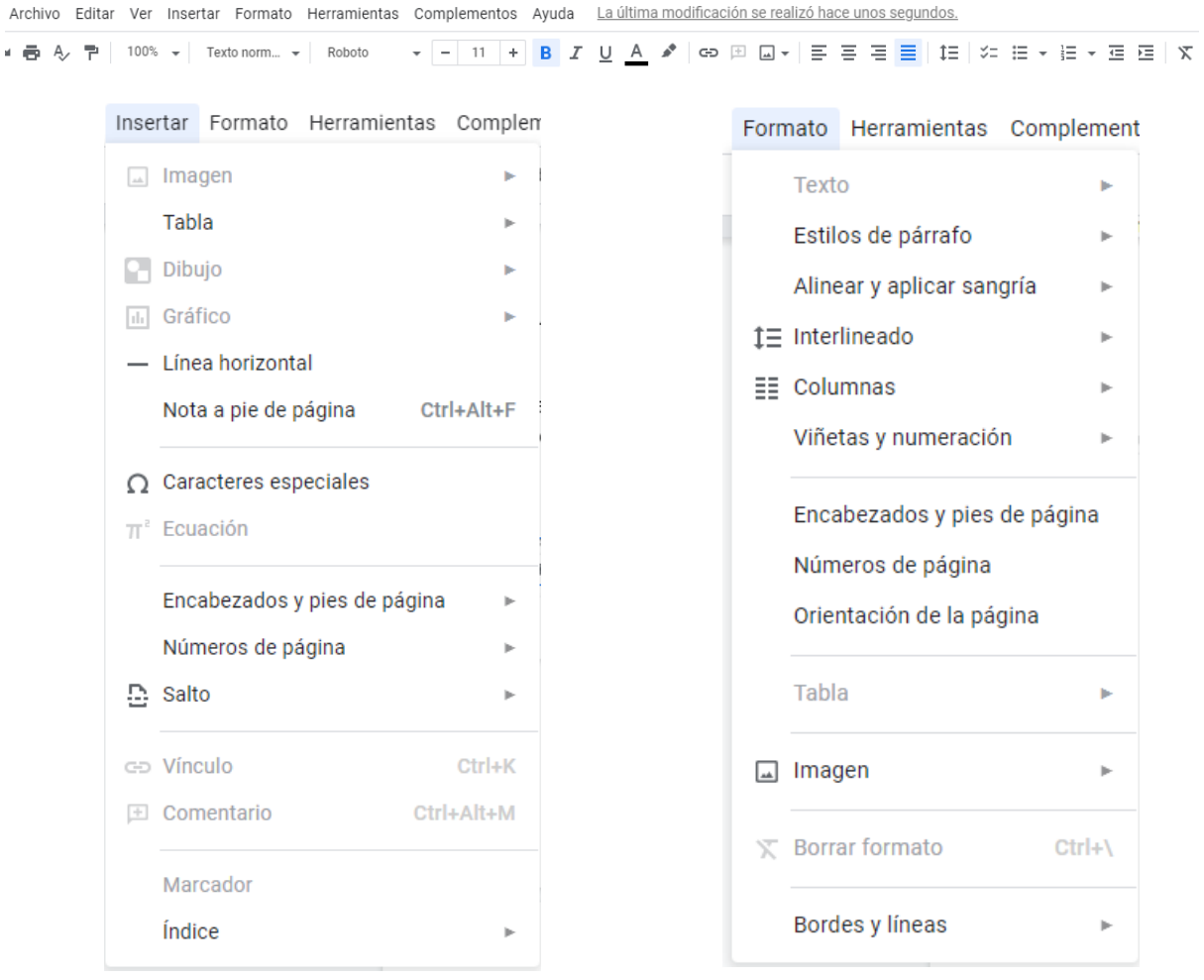
2. Descripción del entorno de desarrollo.

Para este proyecto hemos utilizado distintos entornos de desarrollo, según las tareas que hemos ido realizando.

- **GOOGLE DOCS**

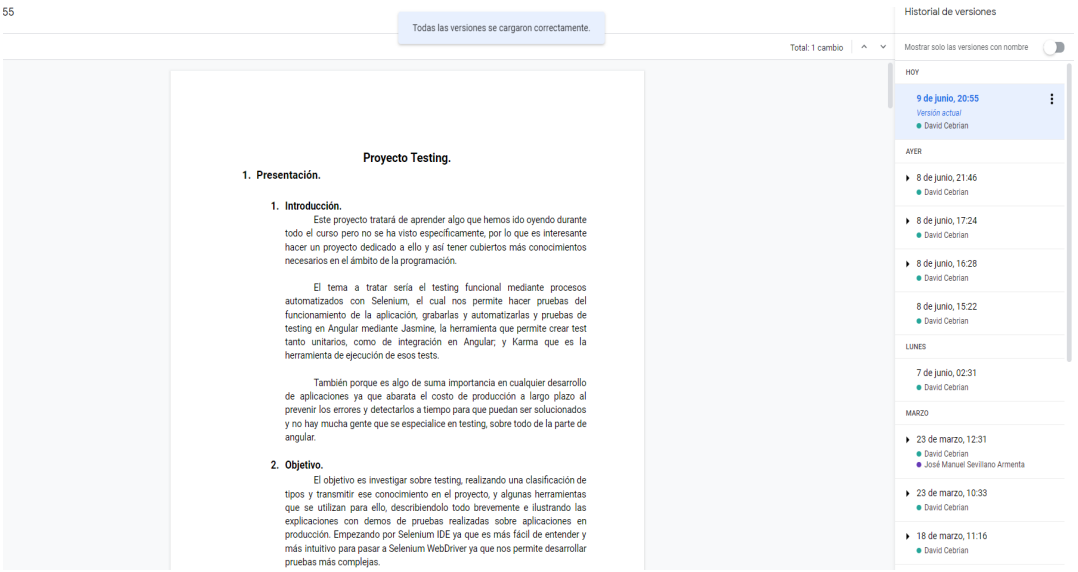
Para el desarrollo teórico hemos utilizado un documento de Word de google drive llamado ***Google Docs***. Este entorno nos ofrece muchas opciones para desarrollar contenido y coordinar versiones con distintos miembros del equipo.

Ofrece las opciones clásicas de los editores de texto, permitiéndonos insertar:



- **Títulos y subtítulos:** Permite indicar cuando un texto es un párrafo , un título o un subtítulo.
- **Listas:** Pueden ser numeradas y no numeradas e incluir sub listas en varios niveles.
- **Imágenes:** Permite insertar gráficos, ilustraciones y cualquier apoyo visual en un archivo de imagen.
- **Tablas:** En “insertar tabla” se puede indicar el número de columnas y líneas para generar rápidamente una tabla.
- **Destacados:** Se pueden usar negritas, cursivas y subrayar palabras y frases para destacarlas, así como cambiar el color y tipo de fuente, la alineación del texto, etc.
- **Trabajo colaborativo:** Permito comentarios en línea, tiene un chat para chatear con la gente con la que compartas el documento y permite la edición simultánea de todos los miembros.
- **Elementos de navegación:** Nos permite insertar hipervínculos con enlaces hacia páginas web, o movernos entre páginas, y también podemos generar un índice según los títulos que hayamos añadido.

- **Historial de versiones:** También nos permite llevar un control de un historial de versiones según los cambios que se vayan guardando, pudiendo volver a versiones anteriores si fuese necesario:



- **SELENIUM IDE**

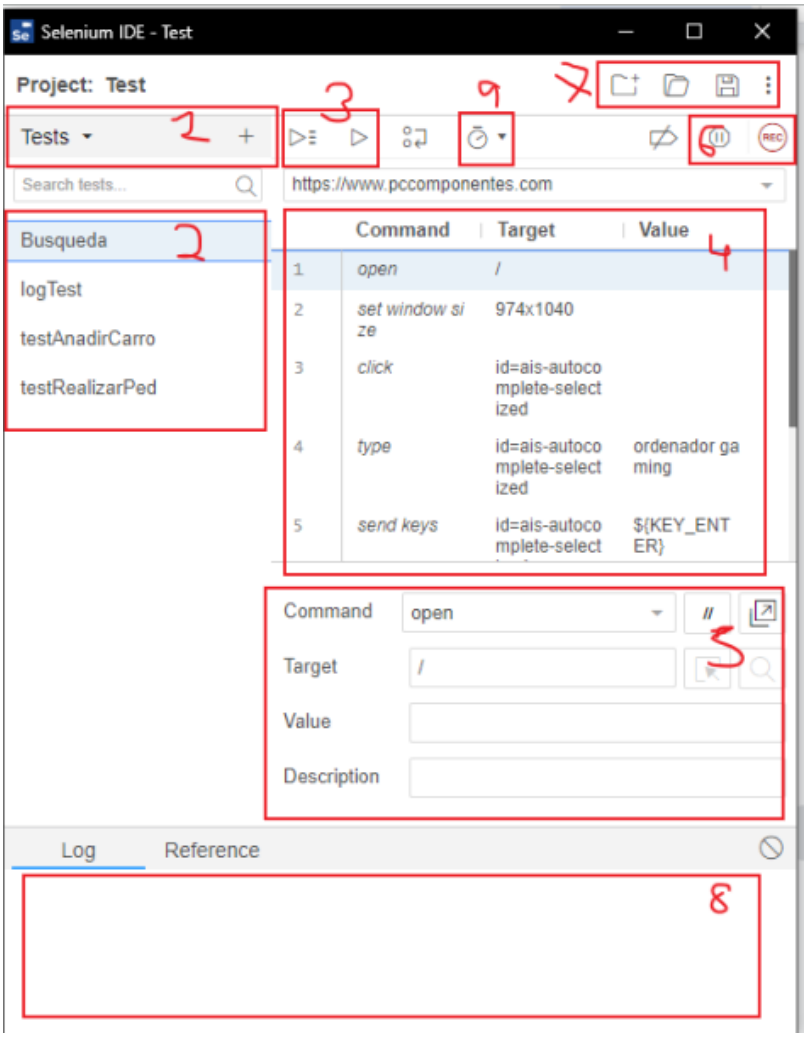
  Es una extensión de Firefox y Chrome que permite escribir test de Selenium con las interacciones del usuario y ejecutarlos directamente desde el navegador.

  Puedes indicarle rutinas de navegación para luego ejecutarlas una y otra vez y detectar así, de una manera sencilla, posibles errores. Todo esto sin necesidad de tener conocimientos de ningún lenguaje de scripting de prueba..

  Este entorno logra desarrollar scripts automáticamente al crear una grabación de las acciones que el usuario va realizando en la página web indicada y de esa forma se puede editar el script para reproducir esa grabación de la forma en que queremos, lo que se puede lograr es que se navegue por la aplicación de forma automática, ya sea realizando clicks, rellenando formularios, verificar la existencia de un elemento, etc.

  Google Docs no necesita instalación ya que es una herramienta online que nos ofrece google drive, lo único que necesitamos es una cuenta de google, la cual podemos crear en (https://accounts.google.com/signup/v2/webcreateaccount?flowName=GlifWebSignIn&flowEntry=SignUp)

  Estos scripts son generados en ***Selanese***, un idioma de scripting para Selenium, y este es el entorno:



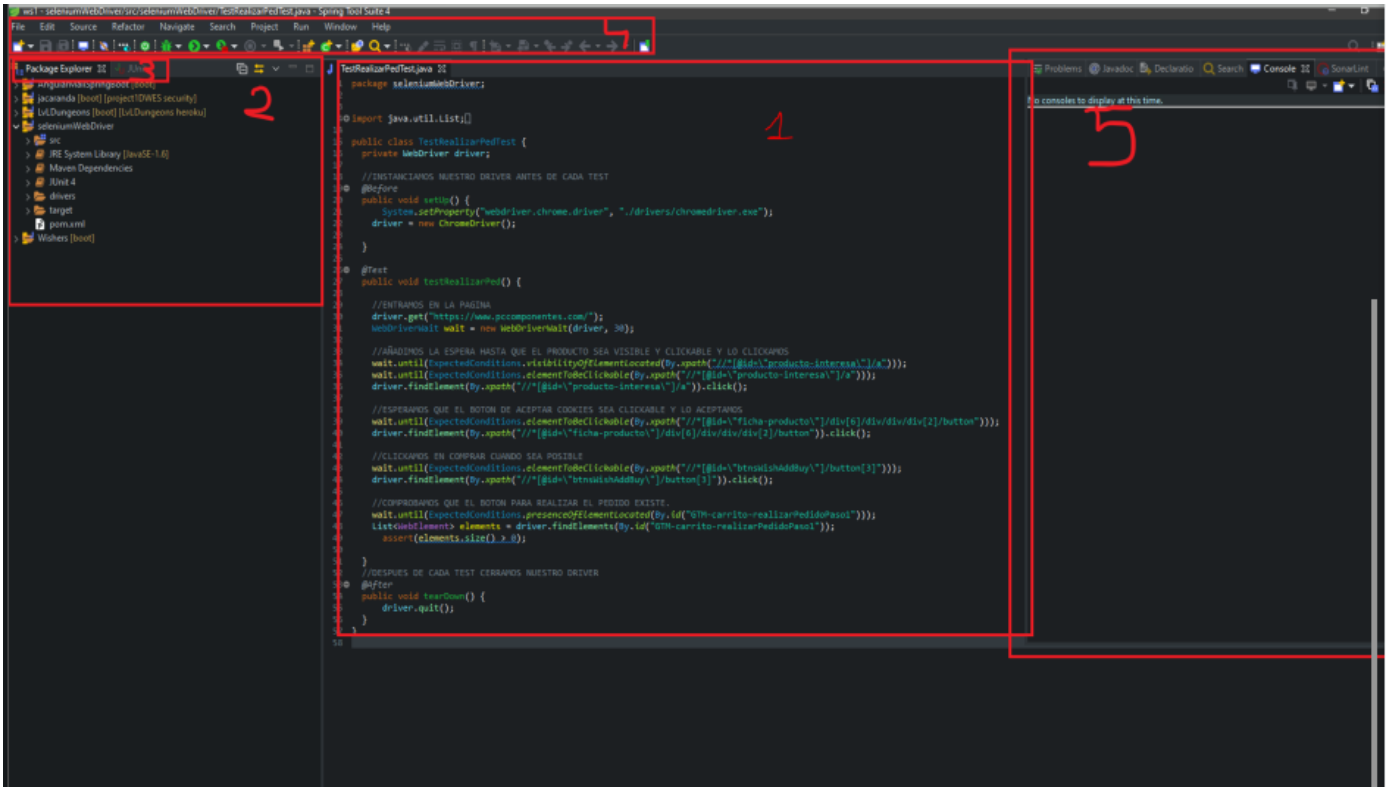
1- Creación de test.
2- Tests creados
3- Ejecutar todos los test, a su lado ejecutar el test seleccionado.
4- Línea de comandos guardados en el script al realizar las acciones.
5- Podemos añadir un nuevo comando a la línea anterior manualmente, como puede ser una espera a que el elemento esté presente.
6- Botón de parar la "grabación" o la captación de las acciones en la página.
7- Botones para guardar los test o abrir un proyecto ya existente.
8- Marco de visualización de logs de salida al ejecutar test.
9- Medidor para añadir tiempo de espera entre la ejecución de un comando y otro.

Su instalación y uso están descritos en el documento Testing Basico (https://github.com/davidcebrian/testingBasico/blob/main/Testing%20Basico.pdf) o mediante el README del github del proyecto:
https://github.com/davidcebrian/testingBasico
En el apartado ***Selenium IDE***. Este es el documento del desarrollo de la teoría mencionada anteriormente sobre Testing.

- **SELENIUM WEB DRIVER**

  Para la realización de las pruebas con selenium webdriver, hemos utilizado el entorno **Spring Tool Suite** (**Eclipse**) al ser selenium una librería de la cual podemos importar en cualquier entorno para el lenguaje que estemos utilizando, y lo hemos utilizado con Spring ya que nos permite crear proyectos maven e incluir las dependencias de una forma mucho más sencilla. El lenguaje utilizado en el entorno ha sido **Java 14** , y hemos añadido aparte de las librerías de selenium las librerías de **JUnit 4** para poder ejecutar esos test.

  Este es el entorno:



1- Marco de desarrollo, dónde escribimos las líneas de código en un determinado lenguaje.

2- Marco de trabajo, donde ubicamos todos los proyectos que vamos creando.

3- Podemos cambiar la pestaña a JUnit para poder ver los resultados de la ejecución de los test.

4- Barra de tareas mediante la que podemos realizar distintas acciones de edición y de configuración tanto del entorno de desarrollo como del proyecto que queramos.

5- Consola. Nos muestra los resultados de las ejecuciones de nuestro código.

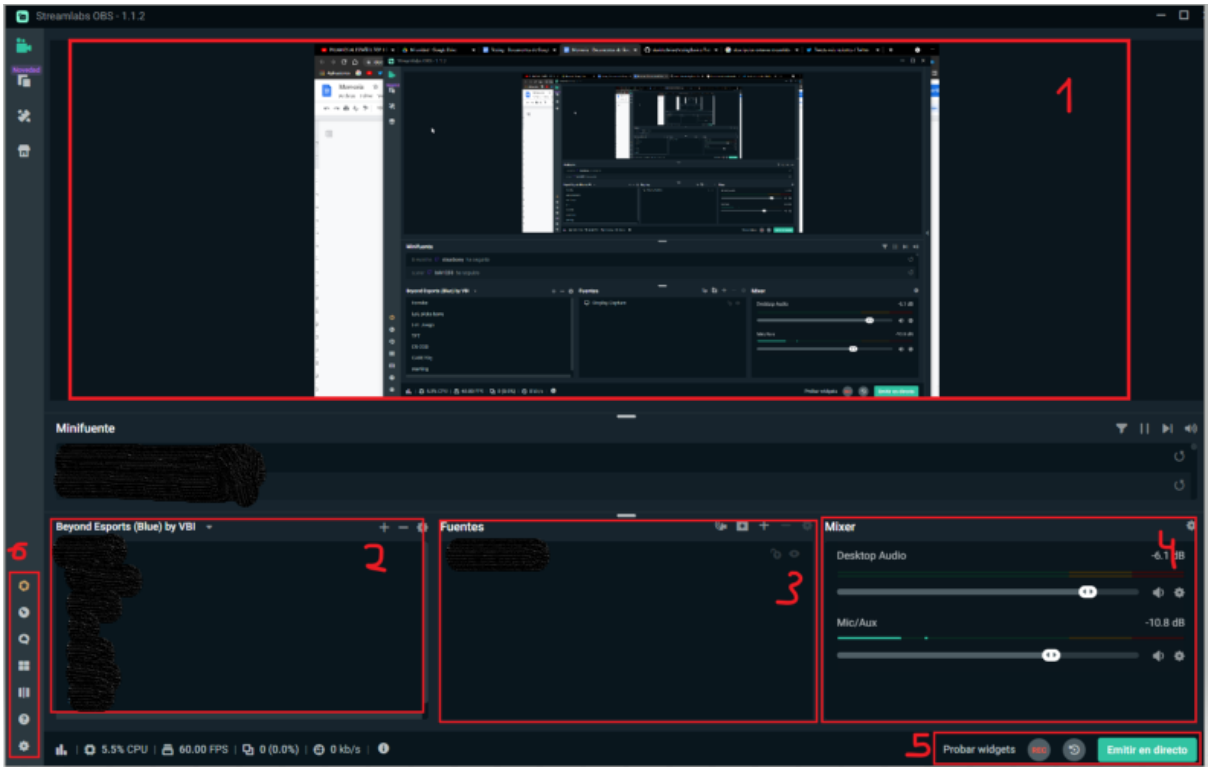
La forma de añadir la librería de selenium está descrita en el documento Testing Básico (https://github.com/davidcebrian/testingBasico/blob/main/Testing%20Basico.pdf) o mediante el README del github del proyecto: https://github.com/davidcebrian/testingBasico en el apartado **SELENIUM WEB DRIVER**

Para la grabación y edición de los vídeos de las pruebas se han utilizado otros dos entornos:

- **STREAMLABS OBS**

  Es un entorno de captura de pantalla o aplicación para poder realizar grabaciones de las misma o retransmitir en directo en distintas aplicaciones como pueden ser Youtube, Twitch o Facebook.

  Tiene una interfaz bastante amigable y fácil de entender:



1- Previsualización de lo que se está viendo o grabando.

2- Escenas creadas o importadas para stream o grabación.

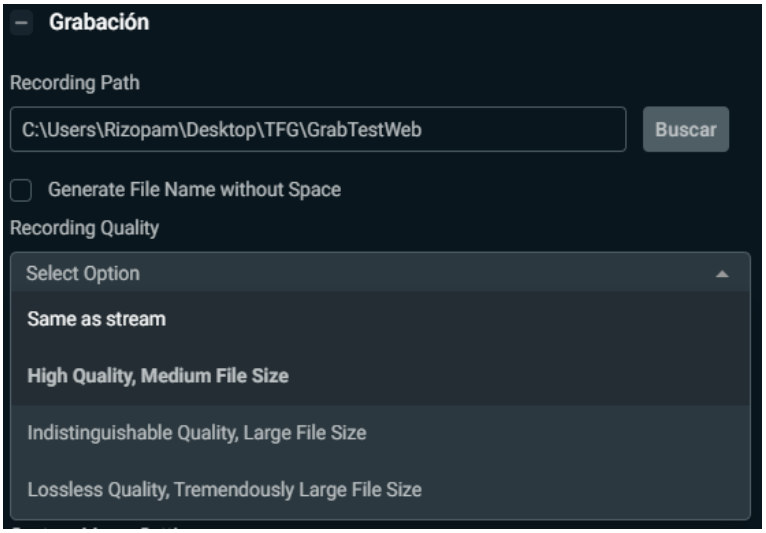
3- Fuentes que podemos ir añadiendo para formar una escena, como puede ser la captura de una aplicación, de una cámara más un texto que se vea en pantalla.

4- Recogida de datos de audio del micrófono y del ordenador.

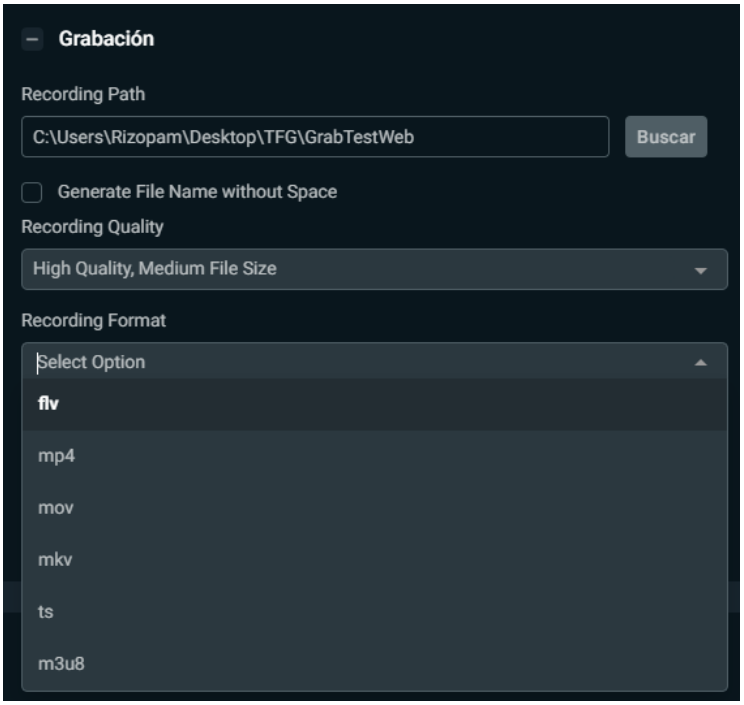
5- Panel de botones para empezar a hacer stream en vivo, o grabar.

6- Panel de botones para acceder a las configuraciones del entorno.

El entorno nos permite grabar la pantalla en distintas calidades:



Y En Distintos Formatos:

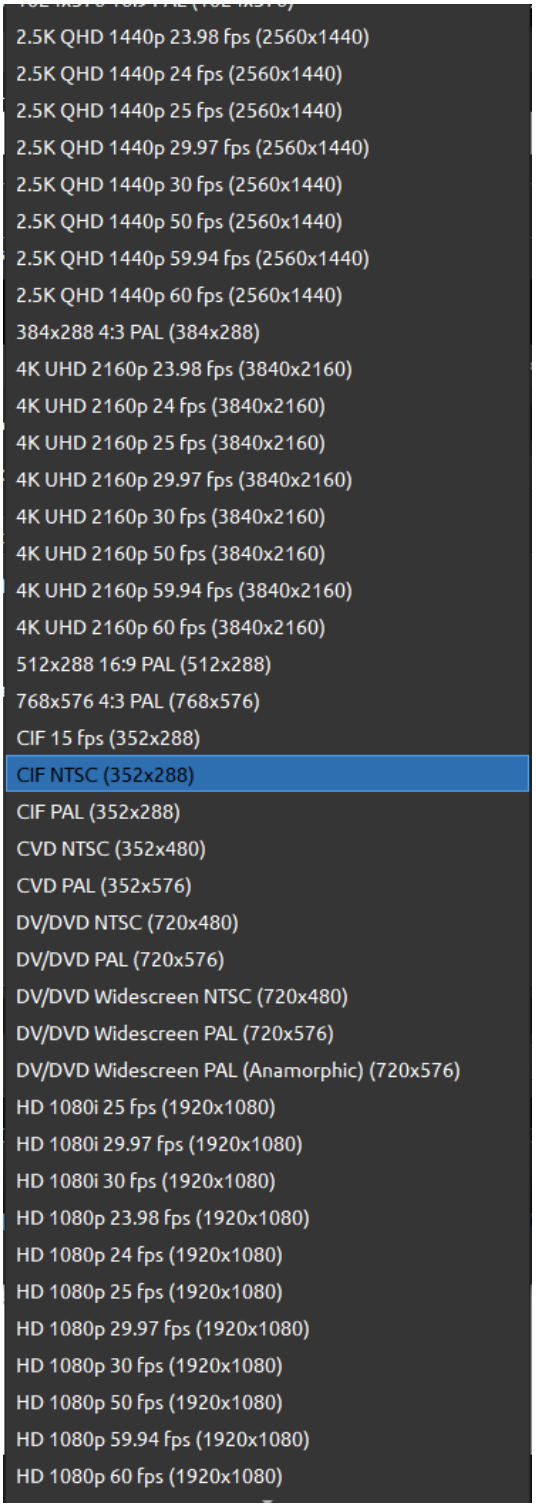


Lo podemos descargar e instalar desde su página oficial:
https://streamlabs.com/?l=es-ES

- **OPENSHOT**

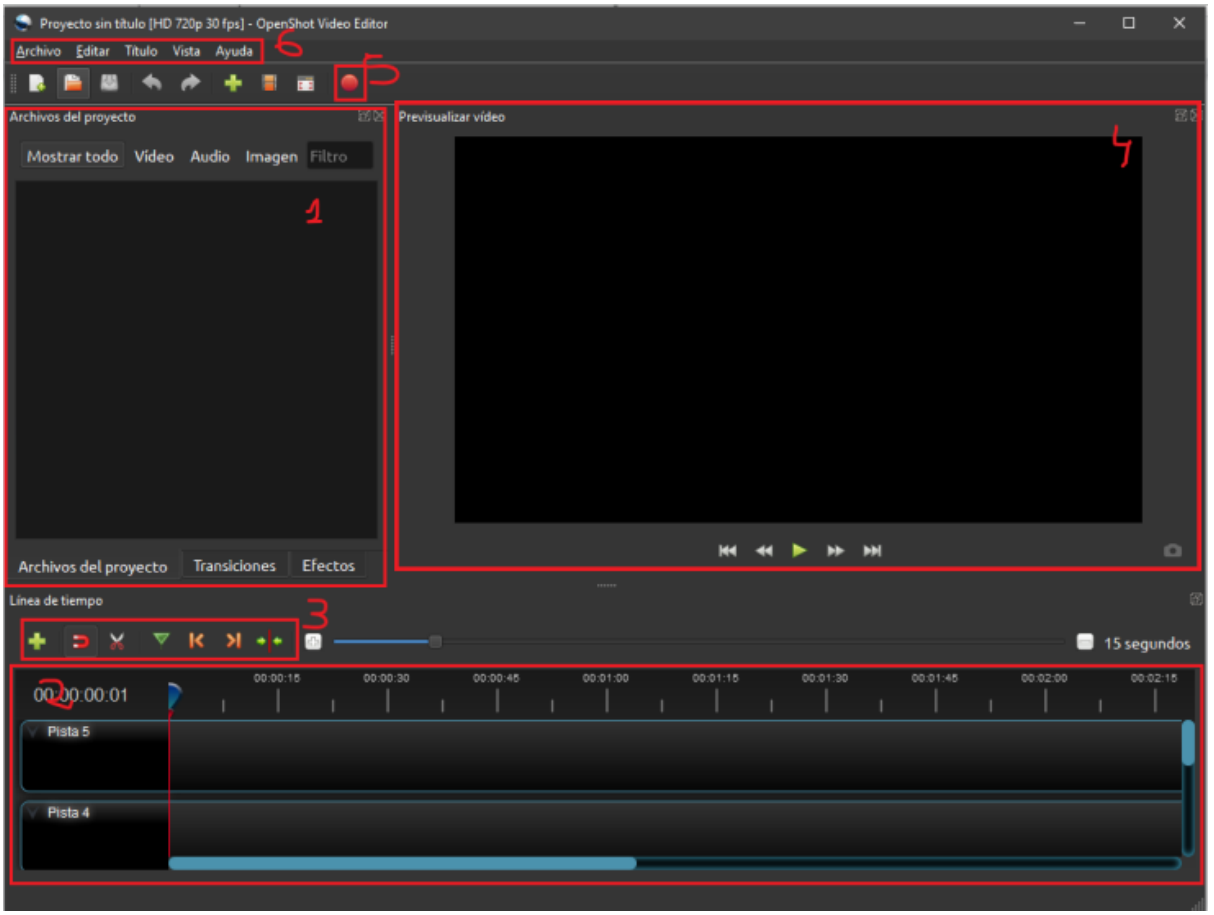
  Este entorno es el utilizado para la edición de los vídeos. Nos permite cortar un vídeo en fragmentos, añadir transiciones, efectos, subtítulos, sonidos, etc.

  Una vez completada la edición podemos exportar el vídeo en distintas formatos:



  Y calidad Alta, Media o Baja.

La interfaz de la aplicación es la siguiente:



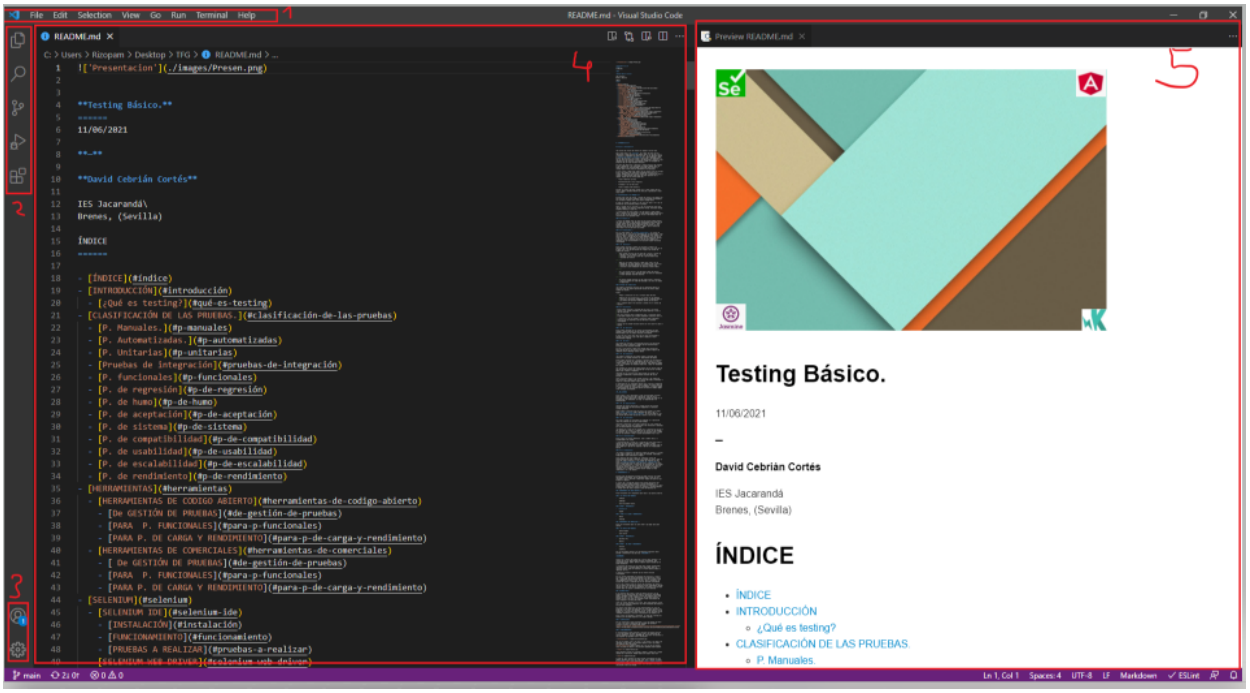
1- Donde ubicamos todos los archivos de nuestro proyecto, así como transiciones, títulos y efectos que queramos añadir.
2- Pistas de edición de los videos y donde añadimos todo lo necesario para la edición del mismo.
3- Marcas para la edición del video sobre la pista.
4- Previsualización del vídeo editado.
5- Exportar el vídeo editado a un formato especificado.
6- Barra de herramientas donde podemos importar archivos o exportarlos, añadir títulos al vídeo, cambiar vistas, etc.

Podemos descargarlo para su instalación desde:
https://www.openshot.org/es/download/

- **VISUAL STUDIO CODE**

  Por último hemos utilizado Visual Studio para pasar el desarrollo teórico a **markdown** para poder mostrarlo como **README.MD** en el GitHub del proyecto:
  https://github.com/davidcebrian/testingBasico

  Visual Studio Code es un IDE el cual nos permite codificar en distintos idiomas dependiendo de los plugins que le instalemos, además de poder previsualizar a tiempo real lo que estemos editando y que sea posible visualizar(HTML, Markdown, etc).



Ésta es su interfaz, en la que podemos encontrar:

1- Barra de herramientas.

2- Barra de herramientas vertical → Archivos o Workspace abierto, búsqueda por texto, enlaces con Git, plugins instalados.

3- Log In y Configuración.

4- Editor de código/texto.

5- Previsualización de archivo **.md**.

Y podemos descargarlo desde:
https://code.visualstudio.com/download

3. DESARROLLO

En cuanto a desarrollo la teoría se ha desarrollado conforme lo mencionado en apartados anteriores. Se ha desarrollado en Google Docs, estilo de letra **Roboto**, tamaño **12** y se han desarrollado todos los apartados mencionados en el índice mostrado en la planificación.

En cuanto a las pruebas realizadas también se mencionan en el documento Testing Básico, están grabadas, editadas y exportadas a calidad **1080p 30fps** para posteriormente ser subidas a la plataforma **YouTube** públicamente en mi canal personal (https://www.youtube.com/channel/UCWTBVVjivf3VdknHDbuhyHA/playlists), añadiendoles en listas de reproducción según el entorno de las pruebas y son las siguientes:

SELENIUM IDE

Las pruebas se realizan sobre la web de pc componentes y el script es grabado en Selanese, el lenguaje en el que se generan las acciones que vamos realizando automáticamente. Pudiendo exportar a otros lenguajes de programación.
(https://www.pccomponentes.com/)

Test Log IN

La primera prueba que realizada ha sido sobre el Login, comprobando una vez que nos logeemos que el elemento en el que aparece nuestra imagen de usuario exista.

1. Entramos en pccomponentes.
2. Pulsamos el avatar de usuarios.
3. Click en log con google.
4. Iniciamos sesión.
5. Esperamos que la letra del usuario esté disponible.
6. Comprobamos que la letra del usuario existe.

Link:
https://www.youtube.com/watch?v=qX4ZkIV3KGg&list=PLdTmqsgulk2X1zRnU94OY4seL3QN_DGbK&index=1

Test Realizar pedido

Otra prueba ha sido comprobar que al clickar en un artículo nos lleva hacia la página de este y desde ahí si clickeamos en comprar nos lleva a la página de realizar el pedido comprobando que existe el botón de esa realización del pedido.

1. Entramos en pccomponentes.
2. Clickeamos en un artículo.
3. Click en comprar.
4. Esperamos que el botón realizar pedido exista.
5. Comprobamos que existe el botón.

Link:
https://www.youtube.com/watch?v=IZgM8NI8Fc8&list=PLdTmqsgulk2X1zRnU94OY4seL3QN_DGbK&index=2

Test Carrito Compra

Comprobamos también que al hacer click sobre la opción de los artículos de añadir al carrito, éstos después existan dentro del carrito de la compra.

1. Entramos en pccomponentes.
2. Clickeamos en un artículo.
3. Click en añadir carrito.
4. Click en el carrito..
5. Comprobamos que existe el botón de realizar compra tras esperar a que exista.

Link:
https://www.youtube.com/watch?v=f1utvQ_1Nv4&list=PLdTmqsgulk2X1zRnU94OY4seL3QN_DGbK&index=1

Test Búsqueda

Comprobaremos que al introducir en la barra de búsqueda algo, busquemos elementos relacionados con lo que hemos escrito. Escribiremos ordenador gaming y comprobaremos que nos ha buscado uno mediante su nombre.

1. Entramos en pccomponentes.
2. Click en la barra de búsqueda.
3. Escribimos ordenador gaming y enviamos.
4. Comprobamos que el primer artículo que sale tiene el texto de un ordenador gaming.

Link:
https://www.youtube.com/watch?v=UJmQntvd3PY&list=PLdTmqsgulk2X1zRnU94OY4seL3QN_DGbK&index=4

SELENIUM WEB DRIVER

Hemos realizado las pruebas sobre la web de pc componentes
(https://www.pccomponentes.com/)

Crear Proyecto.

Explicamos como añadir las dependencias y el driver a un proyecto Maven.

LINK:
https://www.youtube.com/watch?v=Ge5iWw9gdP8&list=PLdTmqsgulk2U_hQ6zZinNaGvNq1hSTIJU

Test Log IN

La primera prueba que realizaremos será sobre el Login, comprobando una vez que nos logeemos que el elemento en el que aparece nuestra imagen de usuario exista.

1. Entramos en pccomponentes.
2. Pulsamos el avatar de usuarios.
3. Extraemos los campos del formulario y enviamos los datos necesarios.
4. Iniciamos sesión.
5. Esperamos que la letra del usuario esté disponible.
6. Comprobamos que la letra del usuario existe.

Link:
https://www.youtube.com/watch?v=XHYbjYbj-gc&list=PLdTmqsgulk2U_hQ6zZinNaGvNq1hSTIJU&index=2

Test Realizar pedido.

Comprobar que al clickar en un artículo nos lleva hacia la página de este y desde ahí si clickeamos en comprar nos lleva a la página de realizar el pedido comprobando que existe el botón de esa realización del pedido.

1. Entramos en pccomponentes.
2. Clickeamos en un artículo.
3. Aceptamos cookies cuando sea posible.
4. Click en comprar.
5. Esperamos que el botón realizar pedido esté disponible.
6. Comprobamos que existe el botón.

Link:
https://www.youtube.com/watch?v=1fnyNdu6sKw&list=PLdTmqsgulk2U_hQ6zZinNaGvNq1hSTIJU&index=5

Test Carrito Compra.

Comprobaremos también que al hacer click sobre la opción de los artículos de añadir al carrito, éstos después existan dentro del carrito de la compra.

1. Entramos en pccomponentes.
2. Clickeamos en un artículo.
3. Aceptamos cookies cuando sea posible.
4. Click en añadir carrito.
5. Comprobamos que existe el botón de realizar compra tras esperar a que aparezca.

Link:
https://www.youtube.com/watch?v=PM_xYH-hKLw&list=PLdTmqsgulk2U_hQ6zZinNaGvNq1hSTIJU&index=4

Test Búsqueda.

Comprobaremos que al introducir en la barra de búsqueda algo, busquemos elementos relacionados con lo que hemos escrito. Escribiremos ordenador gaming y comprobaremos que nos ha buscado uno mediante su nombre.

1. Entramos en pccomponentes.
2. Click en la barra de búsqueda.
3. Escribimos ordenador gaming y enviamos.
4. Comprobamos que el primer artículo que sale tiene el texto de un ordenador gaming.

Link:
https://www.youtube.com/watch?v=uWnUwajy6vE&list=PLdTmqsgulk2U_hQ6zZinNaGvNq1hSTIJU&index=3

4. DOCUMENTACIÓN

1. Instalación de servidores y aplicaciones.

Todo lo relacionado a la instalación de los entornos especificados para el desarrollo de las pruebas y la explicación de sus funciones básicas están explicadas en el documento **Testing Básico**
(https://github.com/davidcebrian/testingBasico/blob/main/Testing%20Basico.pdf)
así como los enlaces dónde podemos encontrar las demás aplicaciones utilizadas se han mencionado en el apartado "**Descripción del entorno de desarrollo**", dónde también describimos sus interfaces para un uso básico.

5. MANTENIMIENTO

Como mantenimiento para este proyecto se propone ir actualizando el documento principal e ir añadiendo grabaciones de lo que se vaya añadiendo al documento al canal de YouTube conforme vayan saliendo actualizaciones de los entornos utilizados que sean de una relevancia importante y sean dignas de mención para un documento destinado a principiantes.

7.POSIBLES MEJORAS

Una de las principales mejoras a este proyecto sería añadir videos de prueba para todos los tipos de test que se han mencionado en él.

Otra posible mejora sería añadir más pruebas distintas con una complejidad mucho mayor para aquellos que ya tengan alguna base puedan expandir un poco más su conocimiento sin tener que buscar la información en otros lugares.

Una gran mejora del proyecto sería recopilar la información recibida junto con los vídeos y montar una página web desde la que acceder a todo sin tener que ir a distintas páginas.

6. CONCLUSIONES.

Este proyecto creo que es algo bueno para las personas que se quieran iniciar en el mundo del testing o quieran hacer alguna prueba con Selenium, ya que se recopila información de distintos sitios y los se intenta plasmar de una forma sencilla para que cualquier persona pueda entenderlo dentro de lo posible.

Así tienen la información básica para comenzar en un solo documento y con explicaciones de Selenium en videos.

Además también me ha ayudado a mí a aprender un poco más sobre el testing al tener que buscar y contrastar información, al tener que hacer algún curso para interiorizar todo aquello que quería comunicar y también me ha servido para aprender a utilizar algunas herramientas como la de grabación y edición que nunca antes había utilizado y ahora tengo unas bases y espero que tanto el documento como el video pueda ayudar a otras personas como lo ha hecho conmigo al realizarlo.

8. REFERENCIAS

La única referencia utilizada para la realización de la memoria ha sido:
http://softwareagil.blogspot.com/2015/04/arquitectura-de-la-prueba-automatizada.html

Las utilizadas en el documento desarrollado “Testing Básico” se mencionan en ese mismo documento.